# MANUAL PASCAL HP4T



**ALPHA Ltd. @ 1991**



# MANUAL PASCAL HP4T



**ALPHA Ltd. @ 1991**

Tipografia MIRTON

1900 TIMISOARA

Str. Samuil Micu nr.7

Telefon 96 - 18.35.25.

CUPRINS

## CITITI RINDURILE DE MAI JOS INAINTE DE A UTILIZA PASCAL.

Inainte de a utiliza Pascal pentru prima data, va recomandam sa adoptati urmatoarea strategie:

1. Cititi Sectiunea 0.0 din acest manual.

2. Cititi Nota de Implementare pentru calculatorul dvs., ZX SPECTRUM, de la pagina 55.

3. Cititi Sectiunea 4 din acest manual si studiati exemplul dat la pagina 43 din aceasta Sectiune.

4. Adaugati linia 195 la programul exemplu:

   195 FOR I:=1 TO Size DO WRITE (Numbers[I]:4)

   si corectati linia 190 astfel incit sa se termine cu ';'.

5. Cititi Sectiunea 0.2 din acest manual, compilati si rulati programul exemplu.

6. Salvati acest program pe caseta cu 'P10,200,example'.

7. Stergeti programul cu 'D10,200'.

8. Reincarcati programul de pe caseta cu 'G,,example'.

9. Compilati si rulati programul din nou.

Daca reusiti sa parcurgeti toti pasii de mai sus cu succes, atunci puteti incepe sa utilizati Pascal, consultind la nevoie manualul. Daca insa intimpinati dificultati in parcurgerea pasilor 1-9, atunci reluati de la pasul 1 si cititi din nou cu atentie Sectiunile la care se fac referiri.

## SECTIUNEA 0 INTRODUCERE.

### 0.0 Punerea in functiune.

Hisoft Pascal 4T (HP4T) este o versiune rapida, usor de folosit si in acelasi timp puternica a limbajului Pascal, asa cum este specificat in Pascal User Manual and Report (Jensen/Wirth Second Edition). Omisiunile fata de aceasta specificatie sint urmatoarele:

FILE nu este implementata, totusi variabilele pot fi stocate pe banda.
Tipul inregistrare nu poate avea variante.
Procedurile si functiunile nu sint valabile ca parametri.

Au fost incluse multe functiuni si proceduri suplimentare, pentru a reflecta situatiile diferite in care sint utilizate calculatoarele; printre acestea sint POKE, PEEK, TIN, TOUT si ADDR.

Compilatorul ocupa aproximativ 12K din memorie, dar in timpul rularii necesita numai circa 4K. Este furnizat pe banda, in format executabil. Toate interfetele intre HP4T si masina gazda au loc prin vectori plasati in mod convenabil la inceputul modulelor executabile (vezi HP4T Alteration Guide). Aceasta permite ca utilizatorul sa poata scrie usor rutinele I/O proprii.

Hisoft Pascal 4T utilizeaza diferite coduri de control, mai ales in cadrul editorului. Desigur, calculatoarele pot avea diferite configuratii de tastatura si astfel diferite moduri de formare a codurilor de caractere. In acest manual, caracterele de control utilizate vor fi RETURN, CC, CH, CI, CP, CS si CX. Notele de implementare indica tastele corespunzatoare sistemului de calcul utilizat.

Ori de cite ori HP4T asteapta la o linie de text in curs de introducere, caracterele de control pot fi folosite dupa cum urmeaza:

| | |
|---|---|
| **RETURN** | se foloseste pentru a termina linia; |
| **CC** | se revine in editor; |
| **CH** | sterge ultimul caracter introdus; |
| **CI** | deplaseaza la urmatoarea pozitie TAB; |
| **CP** | schimba iesirea pe printer (daca exista), iar daca iesirea era pe printer se revine pe ecran; |
| **CX** | sterge intreaga linie. |

In pachetul HP4T este inclus un program simplu de incarcare, cu care utilizatorul poate incarca de pe caseta datele care au fost inregistrate in format HP4T.

Astfel, pentru a incarca compilatorul si rutinele de executie de pe caseta originala furnizata de Hisoft, utilizatorul trebuie sa incarce mai intii programul de incarcare - furnizat in acest scop intr-o forma acceptabila pentru sistemul de operare al calculatorului dvs. Daca utilizatorul nu are acces la sistemul de operare al calculatorului sau, atunci programul de incarcare trebuie sa fie introdus in memoria calculatorului direct, cu ajutorul fie al unui asamblor, fie al unui limbaj de nivel inalt, cum este BASIC. Amanunte despre aceasta operatie, precum si un incarcator tip sint incluse in "HP4T Alteration Guide".

Odata incarcatorul lansat in executie, el va cauta un fisier inregistrat in format HP4T si cind va gasi un asemenea fisier il va incarca in memorie. Daca la citirea benzii este detectata o eroare, va fi afisat un mesaj; trebuie sa readuceti banda la inceputul fisierului si sa incercati sa-l incarcati din nou.

Daca erorile se repeta, ajustati volumul la casetofon.

Cind compilatorul a fost incarcat corect, el va afisa mesajul

Top of RAM?

Trebuie sa raspundeti fie introducind un numar zecimal pozitiv pina la 65535, apoi RETURN, fie direct RETURN (vezi Nota de implementare).

Daca ati introdus un numar, atunci se considera ca acesta reprezinta cea mai mare locatie din RAM+[.?.] si este calculata adresa primei locatii deasupra RAM-ului. Stiva compilatorului este plasata la [.?.], astfel incit puteti rezerva locatiile din memoria inalta (eventual pentru extinderea compilatorului) introducind o valoare mai mica decit RAMTOP-ul real. In versiunea pentru ZX SPECTRUM, 'RAMTOP-ul real' este considerat a fi inceputul zonei de caractere grafice definite de utilizator (UDG in Manualul Sinclair).

Vi se va cere apoi

Top of RAM for 'T'

Puteti introduce fie un numar zecimal, fie implicit valoarea anterioara pentru 'Top of RAM'. Valoarea introdusa va fi considerata drept stiva pentru cazul cind se executa codul obiect rezultat prin comanda 'T' a editorului (vezi Sectiunea 4 pentru amanunte). Este necesar sa definiti o stiva de rulare diferita de RAMTOP-ul real daca, de exemplu, ati scris extensii la modulele de executie, pe care le veti introduce in locatiile de deasupra RAMTOP.

In final, se va cere

Table size?

Ceea ce introduceti va fi marimea zonei de memorie care trebuie alocata tabelului de simboluri al compilatorului. Ca mai inainte, puteti introduce un numar zecimal pozitiv urmat de RETURN, sau numai RETURN, in care caz pentru marimea acestei zone de memorie va fi adoptata o valoare implicita (memoria RAM disponibila impartita la 16). In aproape toate cazurile, valoarea implicita asigura spatiu mai mult decit suficient pentru tabelul de simboluri [.?.]; tabelul de simboluri nu se poate extinde peste adresa `8000 = 32768 zecimal. Daca mentionati ca va avea loc aceasta depasire, vi se va cere din nou sa introduceti 'Top of RAM' si celelalte.
Optional puteti include 'E' inainte de numarul introdus, daca doriti ca editorul intern sa nu fie retinut pentru a fi utilizat cu compilatorul, de exemplu daca folositi in acest scop un editor propriu (vezi HP4T Alteration Guide pentru amanunte).

La acest stadiu, compilatorul si editorul integral (daca a fost retinut) vor fi relocatate dupa tabelul de simboluri, iar executia transferata editorului.

Nota: Semnul ` din acest manual va fi inlocuit cu # (cod 35 = hexazecimal 23, shift '3') in toate sistemele care nu folosesc U.K. ASCII. Numerele precedate de acest simbol sint hexazecimale.

## 0.1 Scopul acestui manual.

Acest manual nu are intentia de a va invata Pascal; daca sinteti incepator in programarea in Pascal, trebuie sa consultati lucrarile care introduc acest limbaj.

Acest manual este un document de referinta care detaliaza particularitatile implementarii realizate de firma Hisoft.

Sectiunea 1 prezinta sintaxa si semantica instructiunilor acceptate de compilator.

Sectiunea 2 prezinta diferitii identificatori predefiniti disponibili in cadrul sistemului Hisoft pentru constante si functiuni.

Sectiunea 3 contine informatii privind diferitele optiuni disponibile ale compilatorului, precum si formatul comentariilor

Sectiunea 4 arata cum trebuie folosit editorul care face parte integranta din HP4T; daca nu doriti sa utilizati acest editor, ci vreti sa interfatati editorul dvs. propriu, consultati HP4T Alteration Guide.
Sectiunile de mai sus trebuie sa fie citite cu atentie de catre toti utilizatorii.

Anexa 1 detaliaza mesajele de eroare generate atit la compilare, cit si la rulare.

Anexa 2 da lista identificatorilor predefiniti si a cuvintelor rezervate.

Anexa 3 prezinta modul de reprezentare interna a datelor in HP4T - utila pentru programatorii care doresc sa intre in detalii.

Anexa 4 contine citeva exemple de programe in Pascal, exemple care trebuie studiate daca simtiti ca aveti dificultati in scrierea programelor cu HP4T.

Anexa 5 prezinta particularitatile de utilizare a versiunii HP4TM

Anexele 6 si 7 contin citeva aplicatii specifice pentru ZX SPECTRUM.

## 0.2 Compilarea si lansarea in executie.

Pentru amanunte cu privire la scrierea, corectarea, compilarea si rularea unui program HP4T folosind editorul integral, se poate vedea Sectiunea 4 din acest manual. Iar daca folositi editorul dvs. propriu, consultati HP4T Alteration Guide.

Odata ce a fost invocat, compilatorul genereaza un listing de forma:

**xxxx nnnn textul liniei sursa**

unde: xxxx este adresa la care incepe codul generat de aceasta linie,
nnnn este numarul liniei, cu zerourile de aliniere suprimate.

Daca o linie contine mai mult de 80 de caractere, atunci compilatorul va introduce caractere 'linie noua', in asa fel incit lungimea unei linii sa nu depaseasca 80 de caractere.

Listarea poate fi dirijata catre printer prin folosirea optiunii P, daca aceasta este acceptata (vezi Sectiunea 3). Puteti opri in orice moment listarea apasind CS; in continuare, cu CC se revine in editor, iar cu oricare alta tasta se porneste din nou listarea.

Daca pe parcursul compilarii este detectata o eroare, atunci va fi afisat mesajul '*ERROR*', urmat de caracterul '^', plasat dupa simbolul care a generat eroarea si de un numar de eroare (vezi Anexa 1). Listarea se va opri; apasati 'E' pentru a reveni in editor si a corecta linia afisata, 'P' pentru a reveni in editor si a corecta linia precedenta (daca exista), sau oricare alta tasta pentru a continua compilarea.

Daca programul se termina incorect (de exemplu, fara 'END'), atunci va fi afisat mesajul 'No more text' si controlul va reveni la editor.

Daca la compilare se depaseste spatiul alocat tabelului de simboluri, va fi afisat mesajul 'No Table Space' si controlul va trece la editor. In mod normal, in acest caz programatorul va salva programul pe caseta, va reincarca compilatorul si va compila din nou, cu o valoare mai mare pentru 'Table size' (vezi Sectiunea 0.0).

Daca compilarea se termina corect dar contine erori, atunci va fi afisat numarul de erori detectate, iar codul obiect va fi sters. Cind compilarea se termina cu succes, apare mesajul 'Run?'. In cazul cind doriti sa rulati imediat programul raspundeti cu 'Y', altfel controlul va fi inapoiat editorului.

Pe parcursul rularii codului obiect pot fi generate diferite mesaje de eroare (vezi Anexa 1). Puteti intrerupe executia cu CS dupa care puteti opri complet rularea cu CC, sau o puteti relua, apasind oricare alta tasta.

## 0.3 Particularitatile declaratiei TYPE.

Diferitele limbaje au modalitati diferite de a se asigura ca utlizatorul nu foloseste un element de date intr-un mod care este inconsistent cu definitia acestuia.

La o extrema este codul masina, unde nu se prevad nici un fel de verificari asupra tipului variabilelor la care se face referiri. Urmeaza apoi limbaje ca 'Tiny Pascal', produs de firma Byte, in

care date de tip caracter, intreg si boolean pot fi amestecate fara a se genera erori. Urcind mai departe, urmeaza BASIC, care face distinctie intre numere si siruri si uneori intre numere intregi si numere reale (folosind, de ex., simbolul '% ' pentru a marca numerele intregi). Urmeaza apoi limbajul Pascal, care merge mai departe, admitind tipuri distincte de date, enumerate de utilizator. In virful ierarhiei (in prezent) se afla un limbaj ca ADA, in care pot fi definite tipuri numerice diferite, incompatibile.

Implementarile Pascal folosesc in principal doua modalitati de intarire a declaratiei TYPE: echivalenta structurala si echivalenta numelor. Pascal 4 Hisoft utilizeaza echivalenta numelor pentru declaratiile RECORD si ARRAY. Consecintele acestui fapt sint clarificate in Sectiunea 1. Aici ne vom limita la un exemplu; presupunem ca doua variabile sint definite dupa cum urmeaza:

```
VAR A : ARRAY['A'..'C'] OF INTEGER;
    B : ARRAY['A'..'C'] OF INTEGER;
```

Vom fi tentati sa credem ca se poate scrie A:=B, dar in Pascal 4 Hisoft aceasta va genera o eroare (*ERROR*10), deoarece prin definitiile de mai sus au fost create doua 'TYPE records' separate. Cu alte cuvinte, utilizatorul nu a decis ca A si B sa reprezinte acelasi tip de date. Ar fi putut face aceasta prin:

```
VAR A,B : ARRAY['A'..'C'] OF INTEGER;
```

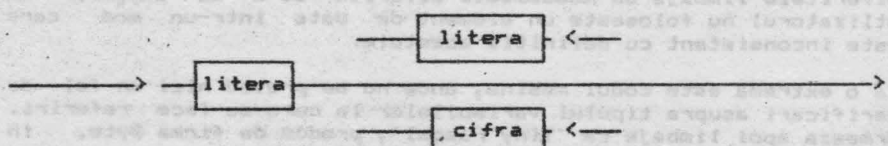
Acum utilizatorul este liber sa atribuie pe A lui B, sau invers, deoarece a fost creat un singur 'TYPE record'.

Desi la prima vedere acest mod de abordare, prin echivalenta de nume pare putin mai complicat, in general duce la mai putine erori de programare, intrucit cere programatorului mai multa gindire initiala.

## SECTIUNEA 1 SINTAXA SI SEMANTICA.

Aceasta sectiune detaliaza sintaxa si semantica limbajului Pascal 4 Hisoft. Daca nu se mentioneaza altfel, aceasta implementare este cea din "Pascal User Manual and Report", Second Edition (Jensen/Wirth).

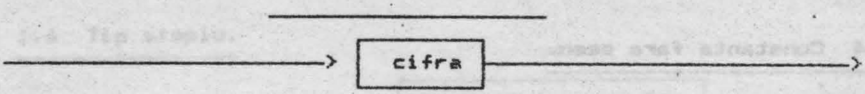
### 1.1 Identificator.



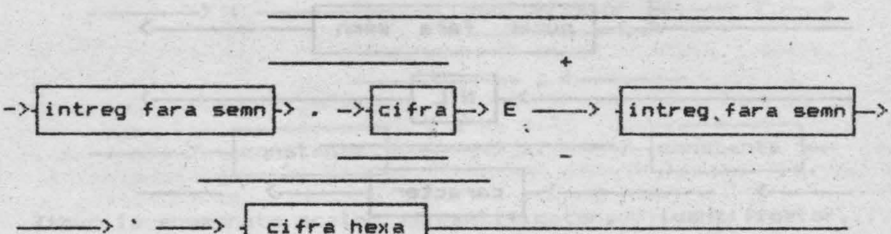
Numai primele 10 caractere ale identificatorului sint tratate ca semnificative.

Identificatorii pot contine litere mari sau mici. Literele mici nu sint convertite in majuscule, astfel incit identificatorii HELLO, HELlo si hello sint diferiti. Cuvintele rezervate si identificatorii predefiniti se introduc numai cu majuscule.

## 1.2 Intreg fara semn.




## 1.3 Numar fara semn.




In Pascal 4 Hisoft intregii trebuie sa aiba o valoare absoluta mai mica sau egala cu 32767. Numerele mai mari sint tratate ca numere reale.

Lungimea mantisei numerelor reale este de 23 de biti. Precizia obtinuta prin utilizarea numerelor reale este de aceea de aproximativ 6 cifre semnificative. De notat ca precizia se pierde daca rezultatul unui calcul este mult mai mic decit valorile absolute ale argumentelor sale; de exemplu, 2.00002 - 2 nu va da 0.00002. Aceasta se datoreste acuratetei cu care fractiile zecimale pot fi reprezentate ca fractii binare. Un asemenea efect nu are loc cind intregi de marime moderata sint reprezentati ca numere reale; de exemplu 200002 - 200000 = 2 exact.

Valoarea maxima a unui numar real este 3.4E38 , iar valoarea minima (absoluta) 5.9E-39.

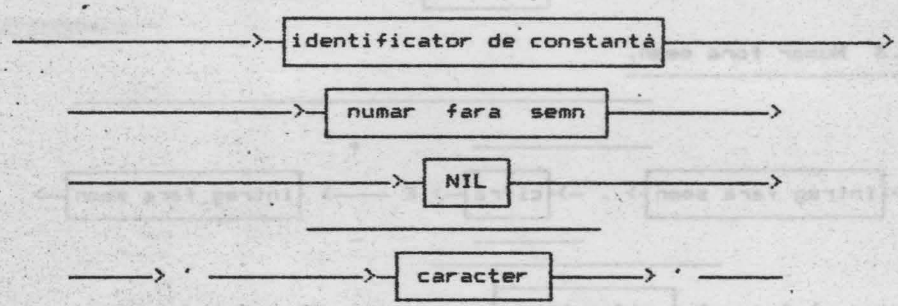
Nu are sens sa se utilizeze mai mult de 7 cifre cind se specifica mantisa unui numar real, intrucit cifrele in plus sint ignorate.

Cind precizia este importanta, evitati zerourile din fata numerelor, intrucit acestea sint numarate ca cifre semnificative.

De exemplu, 0.000123456 este reprezentat mai putin precis decit 1.23456E-4.

Numerele hexazecimale sint disponibile pentru programatori, intre altele pentru a specifica adrese in memorie cind se editeaza legaturi in limbaj de asamblare. De notat ca dupa '`' trebuie sa urmeze cel putin o cifra hexazecimala, altfel va fi generata o eroare (*ERROR*51).

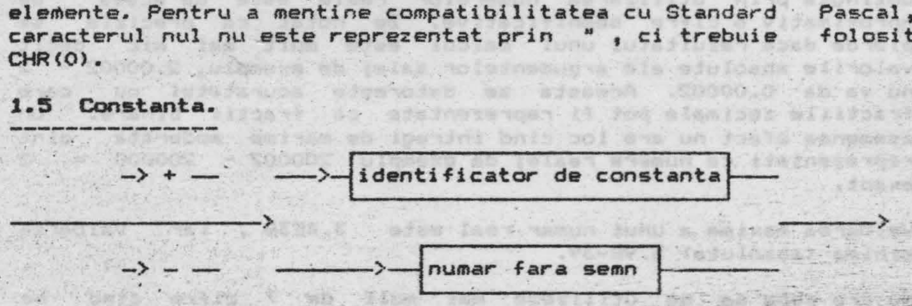
## 1.4 Constanta fara semn.

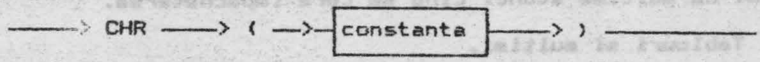



De notat ca sirurile nu pot contine mai mult de 255 caractere. Tipul sir de caractere este definit prin ARRAY[1..N] OF CHAR , unde N este un intreg intre 1 si 255 inclusiv. Sirurile literale nu trebuie sa includa caracterul sfirsit-de-linie (CHR(13)); in caz contrar va fi generat '*ERROR*68'.

Este disponibil intregul set extins de caractere ASCII, cu 256 elemente. Pentru a mentine compatibilitatea cu Standard Pascal, caracterul nul nu este reprezentat prin '' , ci trebuie folosit CHR(0).

## 1.5 Constanta.

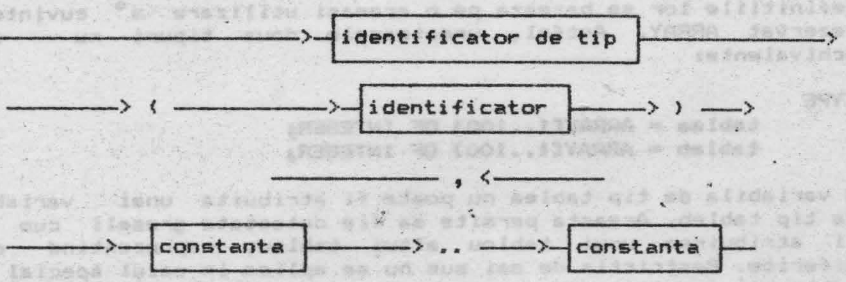





Este prevazuta aici si constructia nestandard CHR, asa incit constantele pot fi utilizate drept caractere de control. In acest caz, constanta dintre paranteze trebuie sa fie de tip intreg.

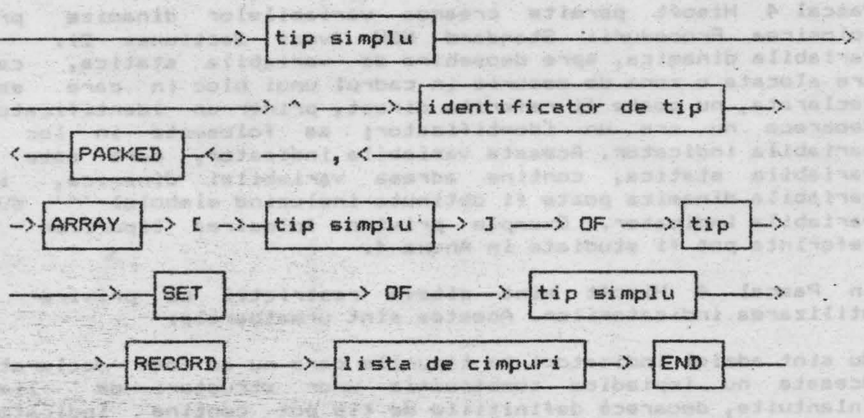
De exemplu: CONST bs=CHR(10);
cr=CHR(13);

## 1.6 Tip simplu.




Tipurile enumerate scalar (identificator, identificator,.....) nu pot avea mai mult de 256 de elemente.

## 1.7 Tip.




Cuvintul rezervat PACKED este acceptat dar ignorat, deoarece impachetarea are deja loc pentru tablouri de caractere etc. Singurul caz in care impachetarea tablourilor ar fi avantajoasa este cel al tablourilor booleene, dar este mai natural de

exprimat ca multime atunci cind se cere impachetarea.

### 1.7.1 Tablouri si multimi.

Tipul de baza al unei multimi poate avea pina la 256 elemente. Aceasta permite ca SET OF CHAR sa fie declarate impreuna cu SETuri ale oricarui tip enumerat, definit de utilizator. De notat, totusi, ca numai subdomenii de intregi pot fi folosite ca tipuri de baza. Orice submultime de intregi este tratata ca multime 0..255.

Sint acceptate tablouri de tablouri, tablouri de multimi, inregistrari de multimi etc.

Doua tipuri de tablouri sint tratate ca echivalente numai daca definitiile lor se bazeaza pe o aceeasi utilizare a cuvintului rezervat ARRAY. Astfel, urmatoarele doua tipuri nu sint echivalente:

```
TYPE
      tablea = ARRAY[1..100] OF INTEGER;
      tableb = ARRAY[1..100] OF INTEGER;
```

O variabila de tip tablea nu poate fi atribuita unei variabile de tip tableb. Aceasta permite sa fie detectate greseli cum ar fi atribuirea unui tablou altui tablou, reprezentind date diferite. Restrictia de mai sus nu se aplica in cazul special al tablourilor de tip sir de caractere, intrucit tablourile de acest tip sint folosite totdeauna pentru a reprezenta date similare.

### 1.7.2 Indicatori.

Pascal 4 Hisoft permite crearea variabilelor dinamice prin folosirea Procedurii Standard NEW (vezi Sectiunea 2). O variabila dinamica, spre deosebire de variabila statica, care are alocata o zona de memorie in cadrul unui bloc in care este declarata, nu poate fi referita direct, printr-un identificator, deoarece nu are un identificator; se foloseste in loc o variabila indicator. Aceasta variabila indicator, care este o variabila statica, contine adresa variabilei dinamice, iar variabila dinamica poate fi obtinuta incluzind simbolul '^' dupa variabila indicator. Exemple privind folosirea tipurilor de referinta pot fi studiate in Anexa 4.

In Pascal 4 Hisoft sint citeva restrictii cu privire la utilizarea indicatorilor. Acestea sint urmatoarele:

Nu sint admisi indicatori la tipurile care nu au fost declarate. Aceasta nu impiedica constructia unor structuri de liste inlantuite, deoarece definitiile de tip pot contine indicatori pentru ele insile, de exemplu

```
TYPE
   item = RECORD
```

```
        value:INTEGER;
        next:^item
        END;

   link = ^item;
```

Nu sint admisi indicatori pentru alti indicatori.

Indicatorii pentru un acelasi tip sint considerati ca fiind echivalenti, de exemplu:

```
VAR
   first:link;
   current:^item;
```

Variabilele first si current sint echivalente (se foloseste adica echivalenta structurala) si ele pot fi atribuite una celeilalte, sau pot fi comparate.

Este acceptata constanta predefinita NIL si daca aceasta constanta este atribuita unei variabile indicator se considera ca aceasta nu contine nici o adresa.

### 1.7.3 Inregistrari.

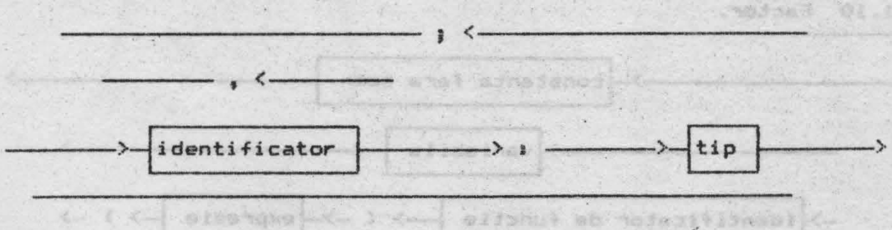
In Pascal 4 Hisoft implementarea inregistrarilor, variabile structurate compuse dintr-un numar fix de constituenti numiti cimpuri, este cea din Standard Pascal, cu exceptia faptului ca nu este acceptata partea variabila a listei de cimpuri.

Doua tipuri de inregistrari sint tratate ca echivalente numai atunci cind declararea lor decurge din o aceeasi aparitie a cuvintului rezervat RECORD (vezi mai inainte, Sectiunea 1.7.1).

Instructiunea WITH poate fi folosita pentru a ne referi la diferite cimpuri din cadrul unei inregistrari intr-o forma mai compacta. Trebuie sa aveti in vedere ca instructiunile WITH nu pot fi apelate recursiv si nu deschid un domeniu nou.

Vezi Anexa 4 pentru un exemplu general de utilizare a lui WITH si RECORD.

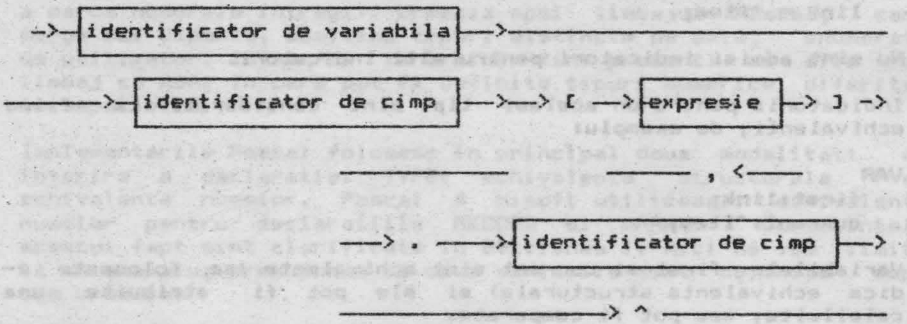
## 1.8 Lista de cimpuri.




Se foloseste in conjunctie cu RECORD (vezi mai inainte

Sectiunea 1.7.4 , precum si exemplul din Anexa 4.).

## 1.9 Variabila.




Pascal 4 Hisoft accepta doua feluri de variabile: variabile statice si variabile dinamice. Variabilele statice sint declarate explicit prin VAR si memoria pentru ele este alocata atunci cind se executa blocul in care au fost declarate.

Variabilele dinamice sint insa create in mod dinamic, in timpul executiei, prin procedura NEW. Ele nu pot fi declarate explicit si nu pot fi mentionate printr-un identificator. Ele sint mentionate indirect, printr-o variabila statica de tip indicator, care contine adresa variabilei dinamice.

Vezi Sectiunea 1.7.2 si Sectiunea 2 pentru mai multe amanunte privind utilizarea variabilelor dinamice, precum si exemplul din Anexa 4.

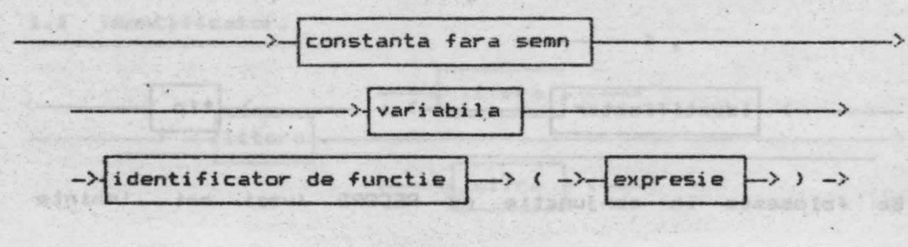
Atunci cind specifica elementele unui tablou multidimensional, programatorul nu este fortat sa foloseasca aceeasi specificatie a indecsilor in referinta ca in declaratie; aceasta este o modificare fata de Pascal 3 Hisoft.

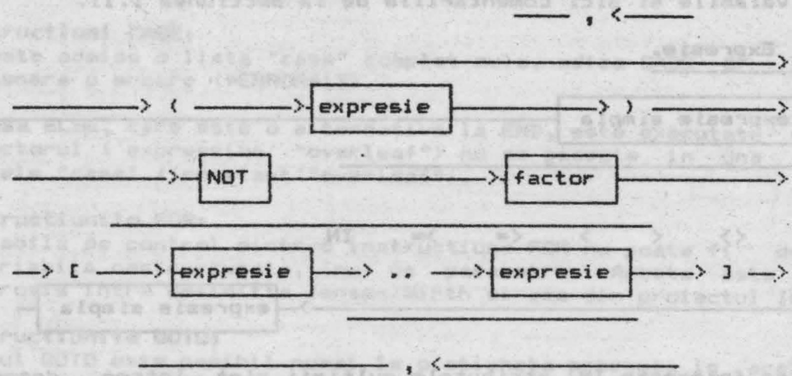
De exemplu, daca variabila a este declarata ca

ARRAY[1..10] OF ARRAY[1..10] OF INTEGER

atunci, pentru a obtine elementul a(1,1) din tablou, se poate folosi fie , fie a[1,1). (?)

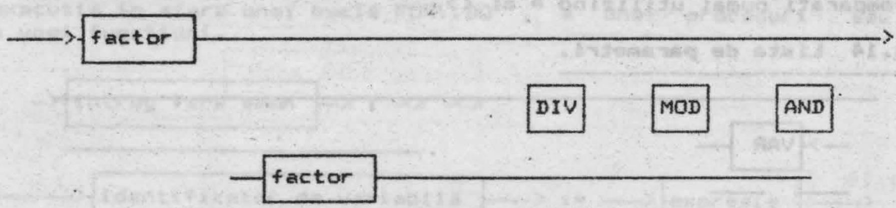
## 1.10 Factor.





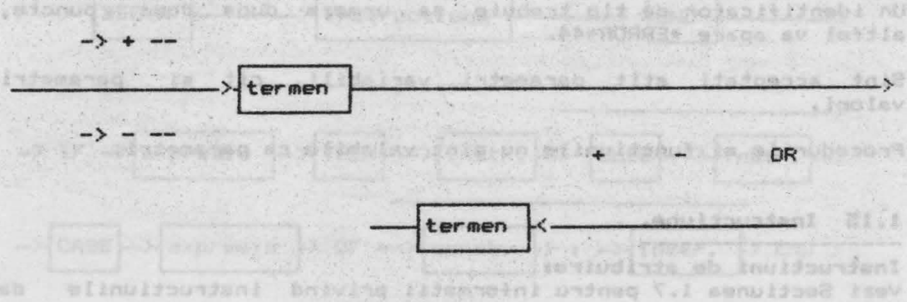

Mai multe amanunte pentru "expresie" in Sectiunea 1.3, iar pentru "functie" in Sectiunea 3.

## 1.11 Termen.



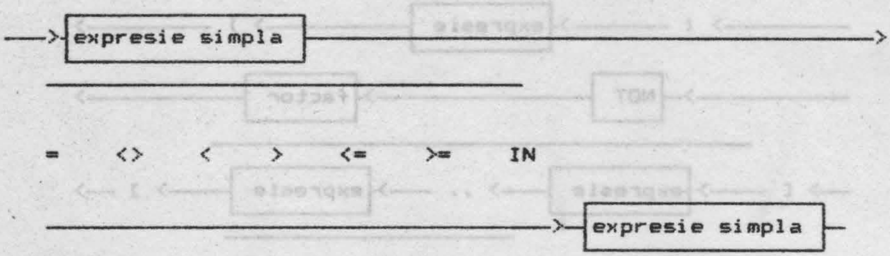

Limita inferioara a unei multimi este totdeauna 0, iar dimensiunea multimii este maximul tipului de baza al [.?.] . Astfel, un SET OF CHAR ocupa totdeauna 32 octeti (256 elemente posibile - un bit pentru fiecare element). In mod similar, un SET OF 0..10 este echivalent cu SET OF 0..255.

## 1.12 Expresie simpla.



Sint valabile si aici comentariile de la Sectiunea 1.11.

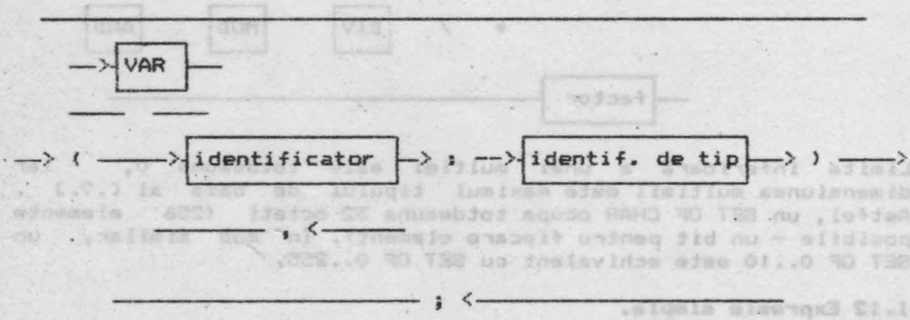
## 1.13 Expresie.




Cind se foloseste IN, atributele multimii sint intreg domeniul tipului expresie simpla, cu exceptia argumentelor intregi, pentru care atributele se iau ca si cum s-ar fi intilnit [0..255].

Sintaxa de mai sus se aplica atunci cind se compara siruri de aceeasi lungime, indicatori si toate tipurile scalare. Multimile se compara folosind >=, <=, <>, sau =. Indicatorii pot fi comparati numai utilizind = si <>.

## 1.14 Lista de parametri.




Un identificator de tip trebuie sa urmeze dupa doua puncte, altfel va apare *ERROR*44.

Sint acceptati atit parametri variabili, cit si parametri valori.

Procedurile si functiunile nu sint valabile ca parametri.

## 1.15 Instructiune.

**Instructiuni de atribuire:**
Vezi Sectiunea 1.7 pentru informatii privind instructiunile de

atribuire care sint ilegale.

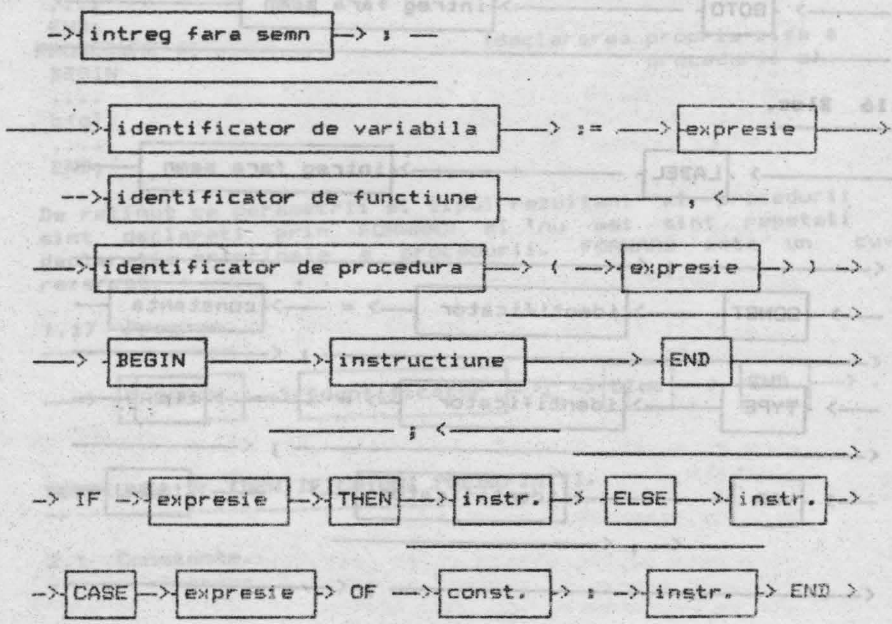
**Instructiuni CASE:**
Nu este admisa o lista "case" complet nula, adica CASE OF END; va genera o eroare (*ERROR*13).

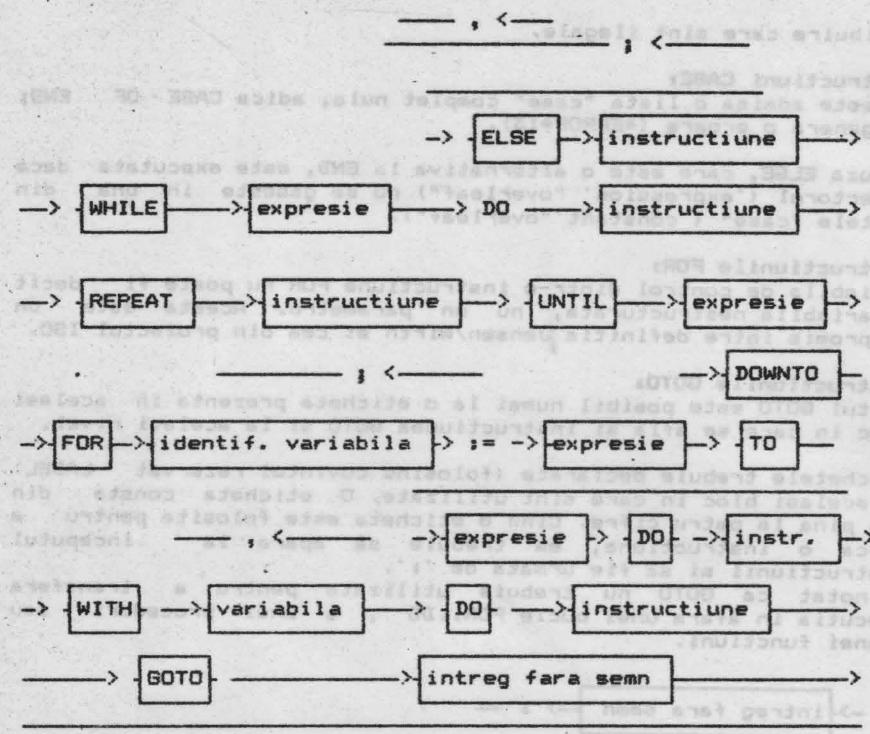
**Clauza ELSE**, care este o alternativa la END, este executata daca selectorul ('expression' "overleaf") nu se gaseste in una din listele "case" ('constant'"overleaf").

**Instructiunile FOR:**
Variabila de control dintr-o instructiune FOR nu poate fi decit o variabila nestructurata, nu un parametru. Acesta este un compromis intre definitia Jensen/Wirth si cea din proiectul ISO.

**Instructiunile GOTO:**
Saltul GOTO este posibil numai la o eticheta prezenta in acelasi bloc in care se afla si instructiunea GOTO si la acelasi nivel.

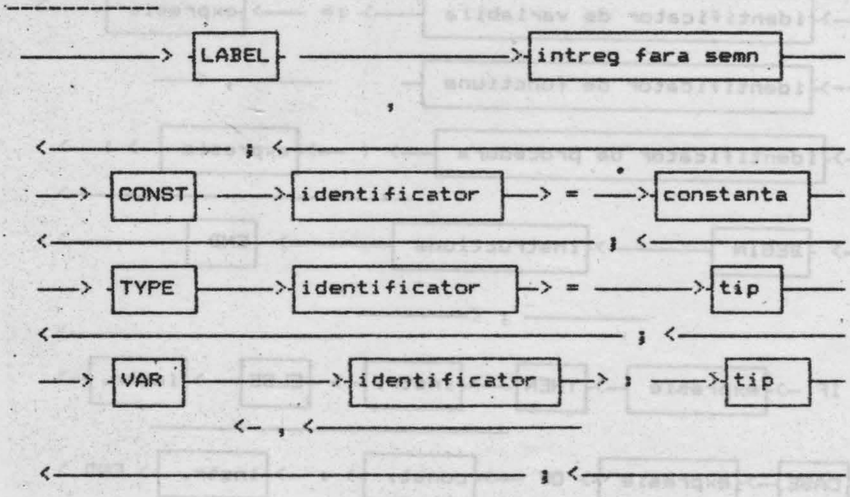
Etichetele trebuie declarate (folosind cuvintul rezervat LABEL) in acelasi bloc in care sint utilizate. O eticheta consta din una pina la patru cifre. Cind o eticheta este folosita pentru a marca o instructiune, ea trebuie sa apara la inceputul instructiunii si sa fie urmata de ':'.
De notat ca GOTO nu trebuie utilizata pentru a transfera executia in afara unei bucle FOR..DO , a unei proceduri sau a unei functiuni.



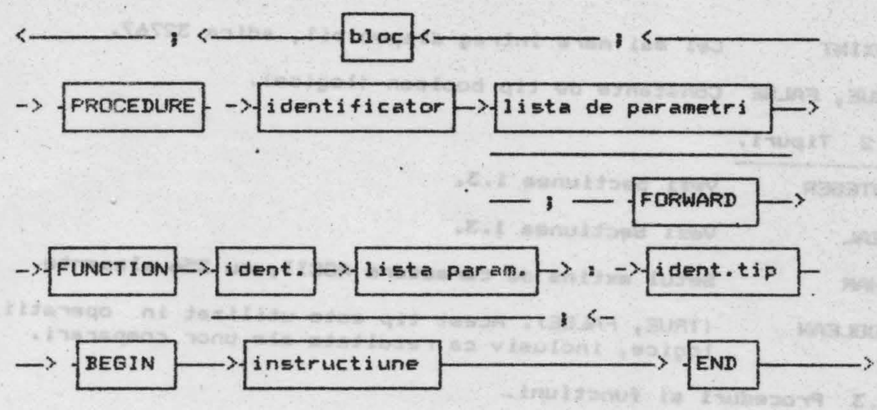


ELSE
instructiune
WHILE
expresie
DO
instructiune
REPEAT
instructiune
UNTIL
expresie
DOWNTO
FOR
identif. variabila
:=
expresie
TO
expresie
DO
instr.
WITH
variabila
DO
instructiune
GOTO
intreg fara semn


## 1.16 Bloc.



LABEL
intreg fara semn
CONST
identificator
=
constanta
TYPE
identificator
=
tip
VAR
identificator
:
tip

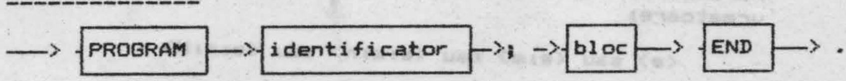



Referirile inainte:
Ca si in Pascal User Manual and Report (Sectiunea 11.C.1), procedurile si functiunile pot fi mentionate inainte de a fi declarate, folosind cuvintul rezervat FORWARD, de exemplu:

```
PROCEDURE a(y:t) ; FORWARD ;    (procedura a este declarata)
PROCEDURE b(x:t);               (inaintea  acestei instructiuni)
 BEGIN:
 ....
 a(p);                          (referire la procedura a)
 ....
 END;
PROCEDURE a;                    (declararea propriu-zisa a
 BEGIN                                   procedurii a)
 ....
 b(q);
 ....
 END;
```

De retinut ca parametrii si tipul rezultant al procedurii a sint declarati prin FORWARD si nu mai sint repetati in declaratia principala a procedurii. FORWARD este un cuvint rezervat.

## 1.17 Program.




# SECTIUNEA 2 IDENTIFICATORI PREDEFINITI.

## 2.1 Constante.

MAXINT Cel mai mare intreg disponibil, adica 32767.

TRUE, FALSE Constante de tip boolean (logice).

## 2.2 Tipuri.

INTEGER Vezi Sectiunea 1.3.

REAL Vezi Sectiunea 1.3.

CHAR Setul extins de caractere ASCII, cu 256 elemente.

BOOLEAN (TRUE, FALSE). Acest tip este utilizat in operatii logice, inclusiv ca rezultate ale unor comparari.

## 2.3 Proceduri si functiuni.

### 2.3.1 Proceduri de intrare si iesire.

#### 2.3.1.1 WRITE

Procedura WRITE este folosita pentru a trimite date catre ecran sau catre printer. Cind expresia care urmeaza sa fie scrisa este simpla, de tip caracter, atunci WRITE(e) transmite catre ecran sau printer, dupa caz, valoarea pe 8 biti a expresiei e.

De retinut:
CHR(8) (CTRL H) muta cursorul inapoi cu un pas pe ecran, cu stergere.
CHR(12) (CTRL L) sterge ecranul, sau executa salt la pagina noua, in cazul printerului.
CHR(13) (CTRL M) executa CR si salt la linie noua.
CHR(16) (CTRL P) comuta, in mod normal, iesirea de la ecran la printer si viceversa.

In general:

WRITE(P1,P2,.......Pn); este echivalenta cu

BEGIN WRITE(P1);WRITE(P2);.......;WRITE(Pn) END;

Parametrii P1, P2,......Pn pot avea una din formele urmatoare:

<e> sau <e:m> sau <e:m:n> sau <e:m:H>

unde e, m si n sint expresii, iar H o constanta literala.

Sint de examinat 5 cazuri:

1) e este de tip intreg si se foloseste <e> sau <e:m>. Valoarea expresiei intregi e este convertita intr-un sir de caractere, cu un spatiu in fata. Lungimea

sirului poate fi marita (cu mai multe spatii in fata) prin folosirea lui m, care specifica numarul total de caractere transmise la iesire.

2) e este de tip intreg si se foloseste forma <e:m:H>.

In acest caz, e este transmis in hexazecimal. Daca m=1 sau m=2, atunci este transmisa valoarea (e MOD 16^m) cu o largime de m caractere. Daca m=3 sau m=4, atunci este transmisa valoarea completa a lui e, in hexazecimal, cu o largime de 4 caractere. Daca m>4, vor fi inserate spatii in fata valorii e complete, in hexazecimal, pina la completarea largimii de m caractere. Daca este cazul, vor fi inserate zerouri la stinga. Exemple:

```
WRITE(1025:m:H)

m=1   afiseaza:  1
m=2   afiseaza:  01
m=3   afiseaza:  0401
m=4   afiseaza:  0401
m=5   afiseaza: _0401
```

3) e este de tip real. Pot fi utilizate formele <e>, <e:m>, sau <e:m:n>.

Valoarea lui e este convertita intr-un sir de caractere care reprezinta un numar real. Formatul reprezentarii este determinat de n. Daca n nu este prezent, atunci numarul este transmis in notatie stiintifica, cu mantisa si exponent. Daca numarul este negativ, este transmis un semn minus inaintea mantisei; in caz contrar este transmis un spatiu. Numarul este transmis totdeauna cu cel putin una si cel mult 5 zecimale, iar exponentul este transmis totdeauna cu semn (plus sau minus). Aceasta inseamna ca largimea minima a reprezentarii stiintifice este de 8 caractere; daca largimea m a cimpului este mai mica decit 8, atunci va fi transmisa totdeauna largimea completa, de 12 caractere. Daca m>=8, vor fi transmise una sau mai multe zecimale (pina la 5 zecimale, cind m=12). Pentru m>12, in fata numarului vor fi inserate spatii. Exemple:

```
WRITE(-1.23E 10:m)

m=7   va da: -1.23000E+10
m=8   va da: -1.2E+10
m=9   va da: -1.23E+10
m=10  va da: -1.230E+10
m=11  va da: -1.2300E+10
m=12  va da: -1.23000E+10
m=13  va da: _-1.23000E+10
```

Daca se foloseste forma <e:m:n>, atunci numarul e va fi

scris in reprezentare cu punct fix, cu n zecimale. Cind largimea m a cimpului este suficient de mare, vor fi transmise in fata spatii. Daca n=0, e va fi transmis ca intreg. Daca e este prea mare pentru a fi scris in largimea specificata de cimp, atunci el va fi transmis in format stiintific, cu un cimp de largime m (vezi mai sus). Exemple:

```
WRITE(1E2:6:2)       va da:  100.00
WRITE(1E2:8:2)       va da:  __100.00
WRITE(23.455:6:1)    va da:  __23.5
WRITE(23.455:4:2)    va da:  _2.34550E+01
WRITE(23.455:4:0)    va da:  __23
```

4) e este de tip caracter, sau tip sir de caractere.

Poate fi folosita forma <e>, sau <e:m> si caracterul, sau sirul de caractere va fi transmis cu o largime minima de cimp egala cu 1 (pentru caractere), sau egala cu lungimea sirului (in cazul sirurilor). Daca m este suficient de mare, vor fi inserate spatii la stinga.

5) e este de tip boolean.

Poate fi utilizata fie forma <e>, fie <e:m> si va fi transmis 'TRUE' sau 'FALSE', in functie de valoarea booleana a lui e, folosind o largime minima de cimp, egala cu 4, respectiv 5.

### 2.3.1.2 WRITELN

WRITELN da la iesire salt la linie noua si CR, fiind echivalenta cu WRITE(CHR(13)). Remarcati ca este inclus salt la linie noua.

WRITELN(P1,P2,.......Pn); este echivalenta cu:

BEGIN WRITE(P1,P2,.......Pn);WRITELN END;

### 2.3.1.3 PAGE

Procedura PAGE este echivalenta cu WRITE(CHR(12)); si determina stergerea ecranului sau salt la pagina noua in cazul iesirii pe printer.

### 2.3.1.4 READ

Procedura READ este folosita pentru a introduce date de la claviatura. Aceasta se realizeaza cu ajutorul unei zone tampon, deschisa pe timpul executarii programului, zona care este initial goala, cu exceptia unui caracter sfirsit-de-linie. Putem considera ca accesul la aceasta zona are loc printr-o fereastra de text care ne permite

sa vedem in zona un singur caracter odata. Daca aceasta fereastra de text este pozitionata in dreptul unui caracter sfirsit-de-linie, atunci, pina ce operatia de citire nu se incheie, va fi citita in zona tampon o noua linie de text de la claviatura. Cind se citeste aceasta linie, vor fi recunoscute diferitele coduri de control mentionate in Sectiunea 0.0.

READ(V1,V2,.......Vn); este echivalenta cu:

BEGIN READ(V1);READ(V2);.......;READ(Vn) END;

unde V1, V2, etc. pot fi de tipul caracter, sir, intreg sau real.

Instructiunea READ(V); are efecte diferite, in functie de tipul lui V. Trebuie considerate 4 cazuri:

1) V este de tip caracter.

In acest caz, READ(V) citeste un caracter din zona tampon de intrare si il atribuie lui V. Daca fereastra din tampon este pozitionata pe un caracter sfirsit-de-linie (un caracter CHR(13)), atunci functia EOLN va primi valoarea TRUE si va fi citita o noua linie de la claviatura. Cind o operatie de citire este ulterior executata, fereastra de text va fi pozitionata la inceputul unei linii noi.

**Nota importanta:** EOLN are valoarea TRUE la pornirea programului. Aceasta inseamna ca daca primul READ este de tipul caracter, atunci va fi returnata o valoare CHR(13), urmata de citirea unei linii noi de la claviatura si urmatoarea citire de tip caracter va returna primul caracter din noua linie, presupunind ca acesta nu este un spatiu. Vezi si procedura READLN mai jos.

2) V este de tip sir de caractere.

Un sir de caractere poate fi citit cu READ si in acest caz vor fi citite o serie de caractere, pina cind se atinge numarul de caractere definit de sir, sau pina cind EOLN=TRUE. Daca sirul nu este completat prin citire (adica daca sfirsit-de-linie este atins inainte ca intregul sir sa fie atribuit), atunci pina la sfirsitul sirului se completeaza cu caractere nule (CHR(0)); in acest fel, programatorul are posibilitatea sa evalueze lungimea sirului care a fost citit.

Nota de la 1) se aplica si aici.

3) V este de tip intreg.

In acest caz sint citite o serie de caractere care

reprezinta un intreg, asa cum acesta este definit in Sectiunea 1.3. Toate spatiile din stinga si markerele sfirsit-de-linie sint ignorate (aceasta inseamna ca intregii pot fi cititi imediat, conform notei anterioare de la 1)).

Daca numarul citit are o valoare absoluta mai mare decit MAXINT (32767), va fi emis mesajul de eroare 'Number too large', iar executia va fi incheiata.

Daca, dupa ce spatiile si caracterele sfirsit-de-linie au fost ignorate, primul caracter citit nu este o cifra sau un semn ('+' au '-'), va fi emis mesajul 'Number expected', iar programul va fi oprit.

4) V este de tip real.

In acest caz vor fi citite o serie de caractere reprezentind un numar real, in acord cu sintaxa din Sectiunea 1.3.

Toate spatiile din fata si markerele sirsit-de-linie sint ignorate si, ca mai inainte, la intregi, primul caracter care urmeaza trebuie sa fie o cifra sau un semn. Daca numarul citit este prea mare sau prea mic (vezi Sectiunea 1.3), va apare eroarea 'Overflow'; daca 'E' este prezent fara sa urmeze un semn sau o cifra, va fi generata eroarea 'Exponent expected'; iar daca este prezent un punct zecimal si dupa el nu urmeaza o cifra, va apare eroarea 'Number expected'.

Numerele reale, ca si numerele intregi, pot fi citite imediat; vezi 1) si 2) mai sus.

### 2.3.1.5 READLN.

```
READLN(V1,V2,.......Vn); este echivalenta cu:
BEGIN READ(V1,V2,.......Vn); READLN END;
```

READLN citeste intr-o zona noua tampon de la claviatura Cind introduceti date in zona tampon, puteti folosi diferitele functii de control mentionate in Sectiunea 0.0. Astfel, EOLN devine FALSE dupa ce se executa READLN daca urmatoarea linie nu este blanc.

READLN poate fi utilizata pentru a sari peste linia blanc prezenta la inceputul executarii codului obiect, adica are efectul citirii intr-o zona noua tampon. Acest lucru este util cind vreti sa cititi la inceputul unui program o componenta de tip caracter, dar nu este necesar daca cititi un numar intreg sau real (intrucit caracterele sfirsit-de-linie sint ignorate), sau daca cititi caractere din liniile ulterioare.

### 2.3.2 Functiuni de intrare.

#### 2.3.2.1 EOLN

Functiunea EOLN este o functiune booleana care inapoiaza valoarea TRUE cind caracterul urmator care se citeste este un caracter sfirsit-de-linie (CHR(13)). Altfel, functiunea inapoiaza valoarea FALSE.

#### 2.3.2.2 INCH

Functiunea INCH executa baleierea claviaturii calculatorului si daca a fost apasata o tasta inapoiaza caracterul reprezentat de aceasta tasta. Daca nu a fost apasata nici o tasta, este returnat caracterul CHR(0). Functiunea inapoiaza asadar un rezultat de tip caracter.

### 2.3.3 Functiuni de transfer.

#### 2.3.3.1 TRUNC(X)

Parametrul X trebuie sa fie de tip real sau intreg iar valoarea returnata de TRUNC este cel mai mare intreg mai mic sau egal cu X, daca X este pozitiv, sau cel mai mic intreg mai mare sau egal cu X, daca X este negativ. Exemple:

TRUNC(-1.5) inapoiaza -1    TRUNC(1.9) inapoiaza 1

#### 2.3.3.2 ROUND(X)

X trebuie sa fie de tip real sau intreg, iar functiunea va returna 'cel mai apropiat' intreg fata de X (in conformitate cu regulile standard de rotunjire). Exemple:

ROUND(-6.5) inapoiaza -6    ROUND(11.7) inapoiaza 12
ROUND(-6.51) inapoiaza -7    ROUND(23.5) inapoiaza 24

#### 2.3.3.3 ENTIER(X)

X trebuie sa fie de tip real sau intreg; ENTIER inapoiaza cel mai mare intreg mai mic sau egal cu X. Exemple:

ENTIER(-6.5) inapoiaza -7    ENTIER(11.7) inapoiaza 11

Observatie: ENTIER nu este o functiune Pascal Standard, insa este echivalenta cu INT din BASIC. Este utila cind se scriu rutine rapide, pentru multe aplicatii in matematica.

2.3.3.4 ORD(X)

X poate fi de orice tip scalar, cu exceptia tipului real. Valoarea returnata este un intreg, reprezentind numarul ordinal al valorii lui X in cadrul setului care defineste tipul lui X.

Daca X este de tip intreg, atunci ORD(X)=X; in mod normal, aceasta utilizare trebuie evitata.

Exemple:

ORD('a') inapoiaza 97    ORD('@') inapoiaza 64

2.3.3.5 CHR(X)

X trebuie sa fie de tip intreg. CHR va returna un caracterul ASCII corespunzator valorii X. Exemple:

CHR(49) inapoiaza '1'    CHR(91) inapoiaza '['

## 2.3.4 Functiuni aritmetice.

In toate functiunile din cadrul acestei subsectiuni, parametrul 'X' trebuie sa fie de tip real sau intreg.

2.3.4.1 ABS(X)

Inapoiaza valoarea absoluta a lui X (de ex. ABS(-4.5) da 4.5). Rezultatul este de acelasi tip ca X.

2.3.4.2 SQR(X)

Inapoiaza valoarea X*X, adica patratul lui X. Rezultatul este de acelasi tip ca X.

2.3.4.3 SQRT(X)

Inapoiaza radacina patrata a lui X, valoarea returnata fiind totdeauna de tip real. Daca argumentul X este negativ, se genereaza 'Maths Call Error'.

2.3.4.4 FRAC(X)

Inapoiaza partea fractionala a lui X:
FRAC(X)=X-ENTIER(X)

Ca si functiunea ENTIER, aceasta functiune este utila pentru scrierea multor rutine matematice rapide. Exemple:

FRAC(1.5) inapoiaza 0.5   FRAC(-12.56) inapoiaza 0.44

2.3.4.5 SIN(X)

Inapoiaza sinusul lui X, X fiind in radiani. Rezultatul este totdeauna de tip real.

2.3.4.6 COS(X)

Inapoiaza cosinusul lui X, X fiind in radiani. Rezultatul este totdeauna de tip real.

2.3.4.7 TAN(X)

Inapoiaza tangenta lui X, X fiind in radiani. Rezultatul este totdeauna de tip real.

2.3.4.8 ARCTAN(X)

Inapoiaza unghiul, in radiani, a carui tangenta este egala cu numarul X. Rezultatul este de tip real.

2.3.4.9 EXP(X)

Inapoiaza valoarea e^X, unde e=2.71828. Rezultatul este totdeauna de tip real.

2.3.4.10 LN(X)

Inapoiaza logaritmul natural (adica in baza e) al lui X Rezultatul este totdeauna de tip real. Daca X<=0, va fi generata 'Math Call Error'.

## 2.3.5 Alte proceduri predefinite.

2.3.5.1 NEW(p)

Procedura NEW(p) aloca spatiu pentru o variabila dinamica. Variabila p este o variabila indicator si dupa ce NEW(p) a fost executata p contine adresa variabilei noi dinamice careia i s-a alocat spatiu. Variabila dinamica este de acelasi tip cu variabila indicator si ele pot fi de orice tip.

Pentru referire la variabila dinamica se foloseste p^. In Anexa 4 este dat un exemplu de utilizare a indicatorilor pentru mentionarea variabilelor dinamice.

Pentru a realoca spatiul folosit pentru variabile dinamice sint folosite procedurile MARK si RELEASE (vezi mai jos).

### 2.3.5.2 MARK(v1)

Aceasta procedura salveaza starea zonei de variabile dinamice in variabila indicator v1. Starea zonei poate fi restabilita la situatia dinaintea executarii procedurii MARK folosind procedura RELEASE (vezi mai jos).

Tipul variabilei indicate de v1 nu are importanta, intrucit v1 poate fi utilizat numai cu MARK si RELEASE, niciodata cu NEW.

Un exemplu de program in care sint folosite MARK si RELEASE este dat in Anexa 4.

### 2.3.5.3 RELEASE(v1)

Aceasta procedura elibereaza spatiu in zona de stocare a variabilelor dinamice. Starea acestei zone este restabilita la starea pe care o avea inainte de a se executa MARK(v1), adica sint distruse toate variabilele dinamice create dupa executarea procedurii MARK(v1). Din aceasta cauza, RELEASE trebuie folosita cu multa precautie.

Mai multe amanunte vezi mai sus, precum si in Anexa 4.

### 2.3.5.4 INLINE(C1,C2,C3,.......)

Aceasta procedura permite ca in programul Pascal sa fie inserat cod masina Z80. Valorile (C1 MOD 256, C2 MOD 256, C3 MOD 256, .......) sint inserate in programul obiect la adresa curenta data de contorul de locatii din cadrul compilatorului. C1, C2, C3 etc. sint constante intregi, numarul lor putind fi oricare. In Anexa 4 este dat un exemplu de utilizare a procedurii

INLINE.

### 2.3.5.5 USER(V)

USER(V) este o procedura cu un singur argument V, intreg. Procedura determina un apel (CALL) la adresa V din memorie. Intrucit Pascal 4 Hisoft memoreaza intregii in format complement fata de 2 (vezi Anexa 3), pentru referiri la adrese mai mari decit `7FFF (32767) trebuie sa folosim valori negative pentru V. De exemplu `C000 este -16384, asa incit USER(-16384); va invoca un apel la adresa `C000. Este, totusi, mai convenabil sa folosim numere hexazecimale atunci cind ne referim la o adresa a memoriei.

Rutina apelata trebuie sa se incheie cu o instructiune Z80 RET (`C9) si trebuie sa conserve registrul IX.

### 2.3.5.6 HALT

Aceasta procedura face ca executia programului sa se opreasca, cu mesajul 'Halt at PC=XXXX', unde XXXX este adresa hexazecimala a locatiei din memorie de unde a fost emis HALT. Impreuna cu un listing de compilare, HALT poate fi folosita pentru a determina, dupa o bifurcare in program pe care din cai continua executia programului. Se foloseste, de obicei, pentru depanarea programelor.

### 2.3.5.7 POKE(X,V)

POKE stocheaza expresia V in memoria calculatorului, incepind de la adresa X. X este de tip intreg, iar V poate fi de orice tip, cu exceptia tipului multime. Vezi discutia de mai sus, de la 2.3.5.5, cu privire la folosirea numerelor intregi pentru a reprezenta adrese in memorie. Exemple:

POKE(`6000,'A') inscrie `41 la locatia `6000.
POKE(-16384,3.6E3) inscrie 00 0B 80 70 (in hexazecimal) la locatia `C000.

### 2.3.5.8 TOUT(NAME,START,SIZE)

TOUT este procedura folosita pentru a salva variabilele pe banda. Primul parametru este de tipul ARRAY[1..8] OF CHAR si este denumirea fisierului care urmeaza sa fie salvat. Sint salvati SIZE octeti de memorie, incepind de la adresa START. Ambii parametri sint de tip intreg. De exemplu, pentru a salva variabila V pe caseta, sub

denumirea 'VAR     ' se foloseste

TOUT('VAR     ',ADDR(V),SIZE(V))

Folosirea adreselor efective din memorie da utilizatorului mult mai multa flexibilitate decit numai abilitatea de a salva tablouri. De exemplu, daca un sistem are ecranul reprezentat in memorie, atunci intreg ecranul poate fi salvat direct. Vezi in Anexa 4 un exemplu de utilizare a procedurii TOUT.

2.3.5.9 TIN(NAME,START)

Aceasta procedura este folosita pentru a incarca de pe banda variabile etc. care au fost salvate cu TOUT. NAME este de tipul ARRAY[1..8] OF CHAR, iar START este de tipul INTEGER. Se cauta pe banda fisierul cu denumirea NAME, care este apoi incarcat in memorie de la adresa START. Numarul de octeti care se incarca este citit in fisier (salvat pe caseta cu TOUT).

De exemplu, pentru a incarca variabila salvata in exemplul de la Sectiunea 2.3.5.8 de mai sus, se foloseste:

TIN('VAR     ',ADDR(V))

Din cauza ca fisierele sursa sint inregistrate de editor in acelasi format cu cel folosit de TIN si TOUT, cu TIN pot fi incarcate fisiere de text in ARRAY OF CHAR, pentru prelucrare ulterioara (vezi HP4T Alteration Guide).

Vezi in Anexa 4 un exemplu de folosire a procedurii TIN

2.3.5.10 OUT(P,C)

Aceasta procedura este utilizata pentru a avea acces direct la portile de iesire ale lui Z80, fara sa mai fie nevoie sa folosim procedura INLINE. Valoarea parametrului intreg P este incarcata in registrul BC, iar parametrul C de tip caracter este incarcat in registrul A, dupa care se executa instructiunea Z80 OUT (C),A.

De exemplu, OUT(1,'A') transmite caraterul 'A' la poarta 1 a microprocesorului Z80.

2.3.6 Alte functiuni predefinite.

2.3.6.1 RANDOM

Aceasta procedura inapoiaza un numar pseudo-aleator cuprins intre 0 si 255 inclusiv. Desi aceasta rutina este foarte rapida, ea da rezultate slabe cind este folosita in mod repetat in bucle care nu contin operatii de intrare-iesire.

Daca utilizatorul are nevoie de rezultate mai bune decit cele pe care le ofera aceasta functie, va trebuie sa scrie o rutina (in Pascal, sau in cod masina), adaptata problemei sale.

**2.3.6.2 SUCC(X)**

X poate fi de orice tip scalar, cu exceptia tipului real, iar SUCC(X) inapoiaza succesorul lui X. De exemplu:

SUCC('A') inapoiaza 'B' SUCC('5') inapoiaza '6'

**2.3.6.3 PRED(X)**

X poate avea orice tip scalar, cu exceptia celui real; rezultatul functiunii este predecesorul lui X. De exemplu:

PRED('j') inapoiaza 'i' PRED(TRUE) inapoiaza FALSE

**2.3.6.4 ODD(X)**

X trebuie sa fie de tipul intreg, iar ODD inapoiaza un rezultat de tip boolean, care este TRUE daca X este impar, sau FALSE daca X este par.

**2.3.6.6 ADDR(X)**

Aceasta functiune utilizeaza un identificator de variabila de orice tip si inapoiaza un rezultat de tip intreg, care este adresa din memorie a identificatorului de variabila V. Pentru informatii privind modul cum sint memorate variabilele in timpul executiei in cadrul implementarii Pascal 4 Hisoft, se poate vedea Anexa 3. Un exemplu de utilizare a functiunii ADDR este dat in Anexa 4.

**2.3.6.7 PEEK(X,T)**

Primul parametru al acestei functiuni este de tip intreg si este folosit pentru a specifica o adresa din

memorie (vezi Sectiunea 2.3.5.5). Al doilea argument este un tip si va fi tipul rezultant al functiunii.

PEEK se foloseste pentru a extrage date din memoria calculatorului si rezultatul poate fi de orice tip.

In toate operatiile PEEK si POKE (opusa lui PEEK), datele sint manipulate in reprezentarea interna proprie pentru Pascal 4 Hisoft, descrisa amanuntit in Anexa 3. De exemplu, daca memoria, incepind de la `5000 in sus, contine valorile 50 61 73 63 61 6C (in hexazecimal), atunci:

```
WRITE(PEEK(`5000,ARRAY[1..6] OF CHAR)) va da 'Pascal'
WRITE(PEEK(`5000,CHAR))  va da P
WRITE(PEEK(`5000,INTEGER)) va da 24912
WRITE(PEEK(`5000,REAL)) va da 2.46227E+29
```

Vezi in Anexa 3 mai multe amanunte privind reprezentarea tipurilor in Pascal 4 Hisoft.

#### 2.3.6.7 SIZE(V)

Parametrul acestei functiuni este o variabila. Rezultatul, de tip intreg, reprezinta numarul de octeti ocupati in memorie de aceasta variabila.

#### 2.3.6.8 INP(P)

INP se foloseste pentru a avea acces direct la portile Z80, fara a mai utiliza procedura INLINE. Valoarea parametrului intreg P este incarcata in registrul BC, iar rezultatul de tip caracter al functiunii se obtine prin executarea instructiunii Z80 IN A,(C).

## SECTIUNEA 3 COMENTARII SI OPTIUNI ALE COMPILATORULUI.

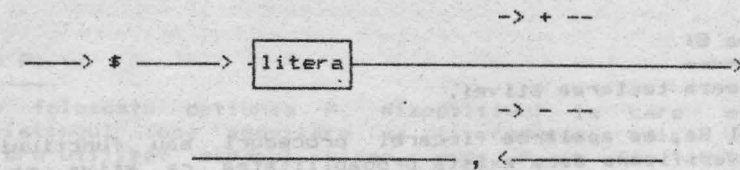
### 3.1 Comentarii.

Un comentariu poate apare intre oricare doua cuvinte rezervate, numere, identificatori sau simboluri speciale (vezi Anexa 3). Comentariul incepe cu caracterul '{', sau cu perechea de caractere '(*'. Toate caracterele care urmeaza - cu exceptia lui '$' - sint ignorate pina la intilnirea unui caracter '}', sau a perechii de caractere '*)'. La aparitia caracterului '$', compilatorul asteapta o serie de optiuni (vezi mai jos), dupa care restul caracterelor sint din nou ignorate, pina apare '}', sau '*)'.

### 3.2 Optiuni ale compilatorului.

---

Sintaxa specificarii optiunilor de compilare este urmatoarea:




Sint disponibile urmatoarele optiuni:

**Optiunea L:**

Controleaza listarea programului text si a adreselor codului masina generat de compilator.

Cind se mentioneaza L+, se face listarea completa.

Pentru L- sint listate numai liniile in care este detectata o eroare.

Implicit: L+

**Optiunea O:**

Controleaza daca se fac unele verificari de depasire. Pentru inmultirea si impartirea numerelor intregi, ca si pentru toate operatiile aritmetice cu numere reale, aceste verificari se fac totdeauna.

In cazul O+ se fac verificari la adunarea si scaderea numerelor intregi.

In cazul O-, aceste verificari nu se fac.

Implicit: O+

**Optiunea C:**

Controleaza testarea claviaturii in timpul executarii programului in cod masina. Daca se specifica C+, atunci executia se opreste cind se apasa CC, cu aparitia unui mesaj HALT - vezi Sectiunea 2.3.5.6.

Aceasta verificare se face la inceputul tuturor buclelor, procedurilor si functiunilor. Utilizatorul poate folosi aceasta facilitate pentru a determina, in cursul unei depanari, care bucla etc. nu se termina corect. Aceasta optiune trebuie evident sa fie dezactivata daca vrem ca programul obiect sa ruleze rapid.

In cazul C-, verificarea de mai sus nu se face.

Implicit: C+

**Optiunea S:**

Controleaza testarea stivei.

In cazul S+, la apelarea fiecarei proceduri sau functiuni se face o verificare daca exista probabilitatea ca stiva sa fie depasita in blocul care urmeaza. Daca exista pericolul ca in timpul executiei stiva sa se suprapuna peste zona afectata variabilelor dinamice sau peste program, executia este oprita si se afiseaza mesajul 'Out of RAM at PC=XXXX'. Evident, nu este certitudine; daca o procedura foloseste mult stiva, atunci programul se poate intr-adevar 'prabusi'. Din contra, in cazul cind o functiune foloseste putin stiva, oprirea executiei se poate dovedi inutila.

In cazul S-, nu se executa verificarea stivei.

Implicit: S+

**Optiunea A:**

Controleaza verificarea incadrarii indicilor tablourilor in limitele specificate la declararea tablourilor respective.

Daca se specifica A+, iar indicele unui element de tablou este prea mare sau prea mic, executia programului este oprita, fiind afisat mesajul 'Index too high', sau 'Index too low'.

In cazul A-, aceasta verificare nu se executa.

Implicit: A+

**Optiunea I:**

Folosind aritmetica numerelor intregi pe 16 biti in complement fata de 2, are loc o depasire cind se executa una din operatiile <, >, <=, >= daca argumentele difera prin mai mult deci MAXINT (32767). Daca acest lucru se intimpla, rezultatul compararii este incorect. In mod normal, aceasta nu supara; daca, insa, utilizatorul doreste sa compare asemenea numere, I+ va asigura corectitudinea rezultatelor. O situatie analoaga poate apare in aritmetica cu numere reale, cind va fi generata o eroare de depasire daca argumentele difera prin mai mult decit aproximativ 3.4E38. O asemenea situatie trebuie evitata.

In cazul I- nu se face verificarea rezultatelor compararilor de

mai sus.

Implicit: I-

**Optiunea P:**

Daca se foloseste optiunea P, dispozitivul la care este trimis listingul dupa compilare va fi comutat, adica daca inainte era utilizat ecranul video, dupa P va fi utilizat printerul si viceversa. Observati ca aceasta optiune nu este urmata de '+', sau '-'.
Implicit: Este utilizat ecranul video.

**Optiunea F:**

Litera F trebuie sa fie urmata de un spatiu si apoi denumirea unui fisier, formata din 8 caractere. Daca denumirea fisierului are mai putin decit 8 caractere, se completeaza cu spatii.

Prezenta acestei optiuni va determina includerea la sfirsitul liniei curente a textului sursa Pascal din fisierul specificat. Este utila in cazul cind programatorul doreste sa alcatuiasca pe banda o 'biblioteca' din multe proceduri si functiuni propri si apoi sa o includa in diferite programe.

Programul trebuie sa fi fost salvat cu comanda 'P' din editor. In majoritatea sistemelor trebuie folosita optiunea L-, altfel viteza de compilare va fi mica.

Exemplu: {$L-,F MATRIX include de pe banda textul fisierului MATRIX};

Cind se editeaza programe foarte mari, s-ar putea sa nu fie loc suficient in memoria calculatorului pentru ca programul sursa si codul obiect sa fie prezente simultan. Este posibil, totusi, sa fie compilate asemenea programe, salvindu-le pe banda si folosind optiunea 'F'; In acest caz, in RAM vor fi - in orice moment - numai 128 octeti din sursa, raminind mult mai mult loc pentru codul obiect.

Aceasta optiune nu poate fi inclusa intr-o bucla si nu este implementata in versiunea pentru ZX Spectrum.

Optiunile compilatorului pot fi utilizate in mod selectiv. Astfel, sectiunile de cod deja depanate pot fi compactate si facute sa ruleze mai rapid, prin dezactivarea verificarilor inutile si retinind aceste verificari numai pentru partile netestate inca ale codului.

## SECTIUNEA 4 EDITORUL INTEGRAL.

## 4.1 Introducere in Editor.

Editorul furnizat cu toate versiunile Pascal 4 Hisoft este un editor simplu, la nivel de linie, proiectat sa lucreze pe toate sistemele de operare Z80, usor de exploatat si putind edita programe rapid si eficient.
Textul este pastrat in memorie intr-o forma compacta; numarul de spatii de la inceputul unei linii este memorat intr-un caracter la inceputul liniei si toate cuvintele rezervate din HP4T sint codificate de asemenea intr-un singur caracter. Aceasta conduce la o reducere tipica a lungimii textului de 25%.

NOTA: in cuprinsul acestei sectiuni, in locul codului de control CH ne vom referi la tasta DELETE.

Editorul este lansat automat in executie dupa ce HP4T este incarcat de pe caseta, afisind mai intii mesajul:

Copyright Hisoft 1982
All rights reserved

si apoi promptul editorului '>'.

Dupa acest prompt, puteti introduce o linie de comanda, cu formatul urmator:

C N1, N2, S1, S2

urmata de RETURN. Aici:

**C** este comanda care urmeaza a fi executata (vezi Sectiunea 4.2 mai jos).
**N1** este un numar in intervalul 1 - 32767 inclusiv.
**N2** este un numar in intervalul 1 - 32767 inclusiv.
**S1** este un sir cu maximum 20 de caractere.
**S2** este un sir cu maximum 20 de caractere.
Virgula este utilizata pentru a separa diferitele argumente (dar acest separator poate fi schimbat, vezi comanda 'S'), iar spatiile sint ignorate, cu exceptia celor din sirurile de caractere. Nici unul dintre aceste argumente nu este obligatoriu desi unele comenzi (cum ar fi comanda 'D'elete) nu actioneaza pina nu sint specificate N1 si N2. Editorul memoreaza numerele si sirurile introduse si, daca nu specificati unul din argumentele liniei de comanda, va utiliza aceste valori introduse anterior, acolo unde este cazul. Initial N1 si N2 au valoarea 10, iar sirurile sint goale. Daca introduceti o linie ilegala de comanda, cum ar fi F-1,100,HELLO, ea va fi ignorata si se va afisa mesajul 'Pardon?'. Linia trebuie introdusa corect adica F1,100,HELLO. Acelasi mesaj de eroare apare si daca lungimea lui S2 depaseste 20; daca lungimea lui S1 depaseste 20, caracterele in plus vor fi ignorate.

Comenzile pot fi introduse cu litere mari sau mici.

La introducerea unei linii de comanda pot fi folosite toate functiile de control descrise in Sectiunea 0.0, de exemplu CX pentru a sterge toata linia.

Subsectiunea urmatoare detaliaza diferitele comenzi disponibile in editor. De retinut ca daca un argument este inchis intre simbolurile '<>' acel argument trebuie sa fie prezent, altfel comanda nu actioneaza.

## 4.2 Comenzile editorului.

### 4.2.1 Inserarea textului.

Textul poate fi introdus intr-un fisier de text fie introducind un numar de linie, un spatiu si apoi textul necesar, fie utilizind comanda 'I'. Retineti ca daca introduceti un numar de linie urmat de RETURN (adica fara text), acea linie va fi stearsa din fisierul de text, daca ea exista. La introducerea unui text pot fi utilizate si functiile de control CX (sterge intreaga linie), CI (salt la urmatoarea pozitie TAB), CC (revenire in bucla de comanda) si CP (comuta pe printer). Tasta DELETE (BACKSPACE) va sterge ultimul caracter introdus si va muta cursorul cu un pas inapoi (dar nu dincolo de inceputul liniei). Textul este introdus intr-o zona tampon interna din HP4T si daca aceasta zona se umple veti fi prevenit, pentru a nu mai introduce text; va trebui, cu DELETE sau CX, sa eliberati memorie in zona tampon.

**Comanda: I n,m**

Cu ajutorul acestei comenzi se obtine intrarea in modul automat de inserare: vor fi afisate numerele de linie, incepind de la n, cu pasul m. Dupa numarul afisat de linie puteti introduce textul necesar, folosind daca doriti diferitele functii de control si terminind linia de text cu RETURN. Pentru a iesi din acest mod se foloseste functia de control CC (vezi Sectiunea 0.0 si Nota de implementare corespunzatoare).

Daca introduceti o linie cu numar de linie care deja exista in text, atunci linia existenta va fi stearsa si inlocuita cu noua linie, dupa ce ati apasat RETURN. Daca incrementarea automata a numarului de linie produce un numar mai mare decit 32767, se va iesi automat din modul Inserare.

Daca, atunci cind introduceti textul, ajungeti la limita ecranului fara sa fi introdus 128 de caractere (marimea zonei tampon), ecranul va defila in sus si puteti continua introducerea pe linia care urmeaza. Textul va capata automat o indentatie, astfel incit numerele de linie sa fie separate efectiv de text.

### 4.2.2 Listarea textului.

Textul poate fi inspectat folosind comanda 'L'; numarul de linii afisat odata la executarea acestei comenzi este fixat initial (vezi Nota dvs. de implementare), dar poate fi modificat cu comanda 'K'.

**Comanda: L n,m**

Aceasta comanda listeaza textul curent pe dispozitivul de afisare, de la linia cu numarul n pina la linia cu numarul m inclusiv. Valoarea implicita a lui n este totdeauna 1, iar a lui m este 32767, adica valorile implicite nu sint luate din argumentele introduse anterior. Pentru a lista intregul fisier de text, se utilizeaza simplu 'L', fara alte argumente. Liniile de ecran sint formate cu o margine in stinga, astfel incit numerele de linie sa fie afisate clar. Dupa listarea unui anumit numar de linii - numar care poate fi controlat cu comanda 'K' - listarea se va opri (daca n-a fost atins numarul de linie m); cu functia de control CC se revine in bucla principala a editorului iar cu oricare alta tasta se continua listarea.

**Comanda: K n**

'K' stabileste numarul liniilor de ecran care urmeaza sa fie listate inainte ca listarea sa fie intrerupta, asa cum este descris la comanda 'L'. Este calculata si stocata valoarea (n MOD 256). De exemplu, folositi K 5 daca vreti ca la 'L'istarea ulterioara sa fie afisate deodata cinci linii de ecran.

## 4.2.3 Editarea textului.

Dupa ce a fost creat un anumit text, in mod inevitabil va fi nevoie de facut corecturi la unele linii. Sint prevazute diferite comenzi care permit ca liniile de text sa fie corectate sterse, mutate sau renumerotate.

**Comanda: D <n,m>**

Toate liniile de la n la m inclusiv sint sterse din fisierul de text. Daca m<n, sau daca sint specificate mai putin de doua argumente nu se intreprinde nici o actiune; aceasta pentru a ajuta la prevenirea erorilor datorate neatentiei. Daca m=n este stearsa o singura linie, dar acest lucru se realizeaza mai simplu scriind numarul liniei urmat de RETURN.

**Comanda: M n,m**

Aceasta comanda face ca textul de la linia n sa fie introdus la linia m, fiind sters orice text care exista inainte aici. Retineti ca linia n ramine neafectata. Asa dar, aceasta comanda va permite sa 'M'utati o linie de text in alta pozitie in cadrul fisierului de text. Daca linia cu numarul n nu exista, nu se va intreprinde nimic.

**Comanda: N <n,m>**

Utilizarea comenzii 'N' are ca efect renumerotarea fisierului de text, prima linie primind numarul n, iar pasul de la un numar de linie la urmatorul fiind m. Ambele numere, n si m, trebuie sa fie prezente in comanda; daca renumerotarea duce la numere de linii mai mari decit 32767, atunci va fi pastrata numerotarea initiala.

**Comanda: F n,m,f,s**

In textul cuprins intre liniile n<m este cautat sirul f. Daca un asemenea sir de caractere este identificat, atunci este afisata linia care il contine si apoi se intra in modul Editare (vezi mai jos). Puteti, in continuare, folosi comenzile din cadrul modului Editare pentru a cauta si alte aparitii ale sirului f in domeniul definit, sau pentru a inlocui aparitia curenta a sirului f cu sirul s, dupa care se cauta aparitia urmatoare a sirului f (vezi mai jos pentru mai multe amanunte).

Retineti ca valorile numerelor de linii si cele doua siruri puteau sa fi fost specificate anterior prin oricare alta comanda si in acest caz este necesar sa introduceti doar 'F' pentru a initia cautarea; vezi exemplul din Sectiunea 4.3 pentru clarificare.

**Comanda: E n**

Editeaza linia cu numarul n. Daca o asemenea linie nu exista, nu se va initia nici o actiune. Daca exista, ea este copiata intr-o zona tampon si afisata pe ecran (impreuna cu numarul de linie). Numarul de linie este apoi afisat din nou, imediat dedesubt si se trece in modul Editor. Toate actiunile ulterioare de editare au loc in zona tampon si nu in textul propriu-zis, astfel incit linia originala poate fi recuperata in orice moment.

In modul Editor, un cursor imaginar se deplaseaza de-a lungul liniei (incepind de la primul caracter); sint disponibile diferite subcomenzi care va permit sa interveniti in textul liniei. Aceste subcomenzi sint urmatoarele:

**(spatiu)** - deplaseaza cursorul cu o pozitie spre dreapta, marcind caracterul urmator din linie. Nu puteti trece cu cursorul dincolo de sfirsitul de linie.

**DELETE** (sau **BACKSPACE**) - deplaseaza cursorul cu o pozitie spre stinga, marcind caracterul anterior. Nu puteti trece cu cursorul dincolo de inceputul de linie.

**CI** (functie de control) - deplaseaza cursorul spre dreapta, pina la urmatoarea pozitie TAB, dar nu dincolo de sfirsitul de linie.

**RETURN** - termina editarea liniei curente, cu pastrarea tuturor modificarilor.

Q - termina editarea liniei curente fara sa retina nici una din modificarile facute, adica lasa linia asa cum era inainte de a se initia corectarea ei.

R - reincarca linia in tamponul editorului, adica ignora toate modificarile facute si reface linia originala.

L - listeaza restul liniei care a mai ramas de corectat, adica partea care urmeaza dupa pozitia curenta a cursorului. Se ramine in modul Editor, cu cursorul repozitionat la inceputul liniei.

K - sterge caracterul care se afla la pozitia curenta a cursorului.

Z - sterge toate caracterele de la pozitia curenta a cursorului (iclusiv) si pina la sfirsitul liniei.

F - gaseste aparitia urmatoare a sirului 'f' de caractere definit anterior printr-o linie de comanda (vezi comanda 'F' mai sus). Aceasta subcomanda va termina automat editarea liniei curente (cu mentinerea modificarilor) daca aici nu este identificata o alta aparitie a sirului 'f'. Daca o aparitie a sirului 'f' este detectata intr-o linie urmatoare (intre limitele specificate anterior), atunci va fi introdus modul Editor pentru linia in care a fost gasit sirul 'f'. Retineti ca, dupa o cautare incheiata cu succes, cursorul va fi pozitionat la inceputul liniei.

S - inlocuieste aparitia curenta a sirului 'f' cu sirul 's' definit anterior, dupa care se executa subcomanda 'F', adica se cauta urmatoarea aparitie a sirului 'f'. Aceasta subcomanda, impreuna cu subcomanda 'F' de mai inainte, este folosita pentru a parcurge fisierul de text si a inlocui, eventual, aparitiile sirului de caractere 'f' cu sirul 's' (vezi Sectiunea 4.3 pentru exemplificare).

**** Nota importanta **** In versiunea curenta a HP4T exista o eroare cunoscuta in functionarea subcomenzii 'S': aceasta subcomanda poate fi utilizata numai imediat dupa o comanda 'F', o subcomanda 'F', sau o subcomanda 'S'. In practica, aceasta insa nu pune probleme.

I - insereaza caractere incepind de la pozitia curenta a cursorului. Se ramine in acest submod pina cind se apasa RETURN; atunci se revine in modul principal Editor, cu cursorul pozitionat dupa ultimul caracter inserat. Folosind DELETE (sau BACKSPACE) in cadrul acestui submod caracterul de la stinga cursorului va fi sters, iar cu CI (functie de control) se va deplasa cursorul in pozitia urmatoare, inserind spatii.

X - avanseaza cursorul la sfirsitul liniei si introduce automat submodul de inserare, descris mai inainte.

C - submodul de modificare. Acesta permite ca peste caracterul aflat la pozitia curenta a cursorului sa se scrie alt caracter, dupa care cursorul avanseaza cu o pozitie. Se ramine in submodul de modificare pina cind se apasa RETURN si atunci se revine in modul Editare, cu cursorul pozitionat dupa ultimul caracter modificat. DELETE (sau BACKSPACE) in cadrul acestui submod muta cursorul cu o pozitie spre stinga iar CI nu are nici un efect.

### 4.2.4 Comenzi pentru lucrul cu banda.

Textul poate fi salvat pe caseta sau incarcat de pe caseta cu ajutorul comenzilor 'P' si 'G'.

**Comanda: P n,m,s**

Liniile din domeniul definit prin n<m sint salvate pe caseta in format HP4T sub denumirea de fisier specificata prin sirul s. Retineti ca aceste argumente pot sa fie specificate si printr-o comanda anterioara. Inainte de a actiona aceasta comanda asigurati-va ca ati pornit casetofonul pe inregistrare. Pe timpul cit se salveaza textul va fi afisat mesajul 'Busy..'.

**Comanda: G,,s**

Pe banda este cautat un fisier in format HP4T, cu denumirea s. Pe timpul cautarii este afisat mesajul 'Busy..'. Daca pe banda a fost gasit un fisier HP4T valabil, dar cu alta denumire va fi afisat mesajul 'Found' urmat de denumirea fisierului gasit, iar cautarea va continua. Odata gasita denumirea corecta de fisier se va afisa mesajul 'Found' si fisierul va fi incarcat in memorie. Daca pe timpul incarcarii este detectata o eroare, va fi afisat mesajul 'Checksum error', iar incarcarea este oprita. Intr-un asemenea caz va trebui sa reinfasurati banda, sa apasati PLAY si sa introduceti din nou comanda 'G'.

Daca sirul s este gol, va fi incarcat primul fisier HP4T intilnit pe banda, indiferent de denumirea lui.

Puteti intrerupe cautarea si incarcarea de pe caseta apasind CS; apasati apoi CC pentru a reveni in bucla editor principala.

Retineti ca daca un fisier de text este deja prezent in memorie fisierul citit de pe banda va fi adaugat la acesta si fisierul rezultat va fi renumerotat, incepind de la numarul de linie 1 si cu pasul 1.

### 4.2.5 Compilarea si lansarea in executie din editor.

**Comanda: C n**

Aceasta comanda va executa compilarea textului care incepe la linia cu numarul n. Daca nu specificati un numar de linie, textul va fi compilat de la prima linie existenta. Pentru mai multe amanunte vezi Sectiunea 0.2.

**Comanda: R**

Codul obiect compilat anterior va fi executat, dar numai daca sursa nu a fost expandata intre timp - vezi Sectiunea 0.2 pentru mai multe detalii.

**Comanda: T**

Este comanda 'T'ranslatare. Textul sursa este compilat de la linia n (sau de la inceput, daca n este omis) si daca compilarea se termina cu succes apare promptul 'Ok?'; daca raspunsul dvs. este 'Y', atunci codul obiect produs prin compilare este deplasat la sfirsitul modulelor de executie (distrugind compilatorul) si apoi modulele de executie si codul obiect vor fi salvate pe banda, denumirea de fisier fiind cea specificata pentru fisierul 'f' definit anterior. Ulterior puteti incarca acest fisier in memorie, folosind incarcatorul HP4T, dupa care automat va fi executat codul obiect. Intrucit codul obiect este deplasat la sfirsitul modulelor de executie, dupa comanda 'T' compilatorul nu va mai fi in memorie si va trebui reincarcat de pe banda. Totusi, aceasta nu ridica probleme, intrucit veti 'T'ranslata un program numai cind este complet functional.

Daca veti decide sa nu continuati cu salvarea pe banda, introduceti oricare alt caracter, afara de 'Y', ca raspuns la promptul 'Ok?'; controlul va reveni la editor, acesta functionind normal, intrucit codul obiect nu a fost deplasat.

**4.2.6 Alte comenzi.**

**Comanda: B**

Aceasta comanda readuce controlul la sistemul de operare. Amanunte asupra modului de reintroducere a compilatorului se gasesc in HP4T Alteration Guide si in Nota dvs. de implementare.

**Comanda: O n,m**

Va amintiti ca textul este stocat in memorie intr-o forma codificata, in care spatiile de la stinga liniei sint concentrate intr-un singur caracter si toate cuvintele rezervate HP4T codificate la cite un singur caracter. Daca, cumva aveti in memorie un text, obtinut cine stie cum, poate cu un alt editor si care nu este codificat, atunci puteti utiliza comanda 'O' pentru a-l codifica. Textul este citit intr-o zona tampon in forma expandata si apoi dus inapoi in fisier sub forma codificata; executia va dura, evident, un oarecare timp. Trebuie specificat un interval de linii, sau, daca nu, vor fi folosite valorile introduse anterior.

**Comanda: S,,d**

Cu aceasta comanda se poate schimba delimitatorul folosit pentru

separarea argumentelor intr-o linie de comanda. La pornirea editorului virgula ',' este luata ca delimitator; aceasta poate fi schimbata, folosind comanda 'S', cu primul caracter din sirul d specificat in comanda. Retineti ca daca ati definit un nou delimitator, acesta trebuie folosit (chiar in cadrul comenzii 'S') pina cind va fi definit un altul.

De notat ca spatiul nu poate servi ca separator.

### 4.3 Un exemplu de utilizare a editorului.

Sa presupunem ca ati introdus urmatorul program (folosind I10,10):

```
 10 PROGRAM BUBBLESORT
 20 CONST
 30   Size = 2000;
 40 VAR
 50   Numbers : ARRAY [1..Size] OF INTEGER;
 60   I, Temp : INTEGER;
 70 BEGIN
 80   FOR I:=1 TO Size DO Number[I] := RANDOM;
 90   REPEAT
100     FOR I:=1 TO Size DO
110     Noswaps := TRUE;
120     IF Number[I] > Number[I+1] THEN
130       BEGIN
140         Temp := Number[I];
150         Number[I] := Number[I+1];
160         Number[I+1] := Temp;
170         Noswaps := FALSE
180       END
190   UNTIL Noswapss
200 END.
```

Acest program contine un numar de erori, dupa cum urmeaza:

Linia 10 Lipseste ';'.
Linia 30 Nu este chiar o eroare, dar sa zicem ca vrem ca marimea sa fie 100.
Linia 100 In loc de 'Size' trebuie sa fie 'Size-1'.
Linia 110 Aceasta linie trebuie mutata la numarul 95.
Linia 190 'Noswapss' trebuie inlocuit cu 'Noswaps'.

In plus, declarata a fost variabila 'Numbers', iar toate referirile sint la 'Number'. In sfirsit, variabila logica 'Noswaps' nu a fost declarata.

## ANEXA 1 ERORI

### A.1.1 Numere de eroare generate de compilator.

1. Numar prea mare.
2. Lipseste ';'.
3. Identificator nedeclarat.
4. Identificator absent.
5. Utilizati '=' si nu ':=' la declararea unei constante.
6. Lipseste '='.
7. Acest identificator nu poate incepe o instructiune.
8. Lipseste ':='.
9. Lipseste ')'.
10. Tip eronat.
11. Lipseste '.'.
12. Lipseste un factor.
13. Lipseste o constanta.
14. Acest identificator nu este o constanta.
15. Lipseste 'THEN'.
16. Lipseste 'DO'.
17. Lipseste 'TO', sau 'DOWNTO'.
18. Lipseste '('.
19. Tip de expresie eronat.
20. Lipseste 'OF'.
21. Lipseste ','.
22. Lipseste ':'.
23. Lipseste 'PROGRAM'.
24. Lipseste o variabila, intrucit parametrul este un parametru variabila.
25. Lipseste 'BEGIN'.
26. Lipseste variabiala intr-o apelare a procedurii READ.
27. Expresiile de acest tip nu pot fi comparate.
28. Tipul trebuie sa fie INTEGER, sau REAL.
29. Acest tip de variabila nu poate fi citit.
30. Acest identificator nu este un tip.
31. Lipseste exponentul unui numar real.
32. Lipseste o expresie scalara (nu numerica).
33. Sirurile vide nu sint admise (folositi CHR(0)).
34. Lipseste '['.
35. Lipseste ']'.
36. Indicele unui tablou trebuie sa fie de tip scalar.
37. Lipseste '..'.
38. In declaratia ARRAY lipseste ']', sau ','.
39. Limita inferioara mai mare decit limita superioara.
40. Multime prea mare (mai mult de 256 elemente admise).
41. Rezultatul functiunii trebuie sa fie identificator de tip.
42. Lipseste ',', sau ']' in multime.
43. Lipseste '..', sau ',', sau ']' in multime.
44. Tipul parametrului trebuie sa fie un identificator de tip.
45. Multimea nula nu poate fi primul factor intr-o instructiune care nu atribuie valori.
46. Lipseste un scalar (inclusiv real).
47. Lipseste un scalar (exclusiv real).
48. Multimi incompatibile.
49. '<' si '>' nu pot fi utilizate pentru a compara multimi.
50. Lipseste 'FORWARD', 'LABEL', 'CONST', 'VAR', 'TYPE', sau 'BEGIN'.
51. Lipseste o cifra hexazecimala.

52. Multimile nu pot apare intr-o functiune POKE.
53. Tablou prea mare (>64K).
54. In definitia unui RECORD lipseste 'END', sau ';'.
55. Lipseste identificatorul de cimp.
56. Lipseste variabila dupa 'WITH'.
57. Variabila din WITH trebuie sa fie de tip RECORD.
58. Identificatorul de cimp nu a fost asociat unei instructiuni WITH.
59. Dupa 'LABEL' trebuie sa urmeze un intreg fara semn.
60. Dupa 'GOTO' trebuie sa urmeze un intreg fara semn.
61. Aceasta eticheta este la un nivel gresit.
62. Eticheta nedeclarata.
63. Parametrul din SIZE trebuie sa fie o variabila.
64. Pentru indicatori sint admise numai testari de egalitate.
67. Pentru intregi, singurul parametru cu doua ':' admis in WRITE este e:m:H.
68. Sirurile nu trebuie sa contina caractere sfirsit-de-linie.
69. Parametrul functiunilor NEW, MARK, sau RELEASE trebuie sa fie o variabila de tip indicator.
70. Parametrul functiunii ADDR trebuie sa fie o variabila.

### A.1.2 Mesaje de eroare in timpul executiei.

Cind o eroare este detectata in timpul executiei, va fi afisat unul din urmatoarele mesaje, urmat de 'at PC=XXXX', unde XXXX este locatia din memorie unde a aparut eroarea. De multe ori sursa erorii este evidenta; daca nu, consultati listingul de compilare pentru a vedea in program unde a aparut eroarea, luind ca referinta adresa XXXX. Uneori aceasta metoda s-ar putea sa nu dea rezultatul corect.

Halt
Overflow
Out of RAM
/ by zero (poate fi generat si de DIV).
Index too low
Index too high
Maths Call Error
Number too large
Number expected
Line too long
Exponent expected

La aparitia acestor erori executia programului se opreste.

## ANEXA 2 CUVINTE REZERVATE SI IDENTIFICATORI PREDEFINITI.

### A.2.1 Cuvinte rezervate.

| | | | | | |
|---|---|---|---|---|---|
| AND | ARRAY | BEGIN | CASE | CONST | DIV |
| DO | DOWNTO | ELSE | END | FORWARD | FUNCTION |
| GOTO | IF | IN | LABEL | MOD | NIL |
| NOT | OF | OR | PACKED | PROCEDURE | PROGRAM |

```
RECORD   REPEAT   SET     THEN    TO     TYPE
UNTIL    VAR      WHILE   WITH
```

### A.2.2 Simboluri speciale.

```
+     -     *     /
=     <>    <     <=    >=    >
(     )     [     ]
{     }     (*    *)
^     :=    .     ,     ;     :
'     ..
```

### A.2.3 Identificatori predefiniti.

Urmatoarele entitati pot fi considerate ca fiind declarate intr-un bloc comun intregului program, fiind de aceea disponibile peste tot in program, daca nu sint redefinite de programator in cadrul unui bloc.
Pentru alte informatii vezi Sectiunea 2.

| | |
|---|---|
| CONST | MAXINT = 32767; |
| TYPE | BOOLEAN = (FALSE, TRUE); CHAR (Setul de caractere ASCII extins); INTEGER = -MAXINT..MAXINT; READ (Submultime a numerelor reale, vezi Sectiunea 1.3.) |
| PROCEDURE | WRITE; WRITELN; READ; READLN; PAGE; HALT; USER; POKE; INLINE; OUT; NEW; MARK; RELEASE; TIN; TOUT; |
| FUNCTION | ABS; SQR; ODD; RANDOM; ORD; SUCC; PRED; INCH; EOLN; PEEK; CHR; SQRT; ENTIER; ROUND; TRUNC; FRAC; SIN; COS; TAN; ARCTAN; EXP; LN; ADDR; SIZE; INP; |

## ANEXA 3 REPREZENTAREA SI STOCAREA DATELOR.

### A.3.1 Reprezentarea datelor.

Discutia care urmeaza da detalii asupra modului cum sint reprezentate intern datele in Pascal 4 Hisoft.

Informatii cu privire la cantitatea de memorie necesara in fiecare caz aparte este utila tuturor programatorilor (poate fi folosita si functiunea SIZE, vezi Sectiunea 2.3.6.7); alte detalii pot fi necesare celor care ar vrea sa includa programe cod masina in programe Pascal.

### A.3.1.1 Intregi.

Intregii ocupa fiecare cite 2 octeti de memorie, in format complement fata de 2. Exemple:

```
   1  =  `0001
 256  =  `0100
-256  =  `FF00
```

Registrul Z80 standard pentru manipularea intregilor este HL.

### A.3.1.2 Caractere, valori logice si alte valori scalare.

Acestea ocupa un octet de memorie fiecare, in format binar pur, fara semn.

Caractere: este folosit codul ASCII extins, pe 8 biti; de exemplu:

```
'E' = `45
'[' = `5B
```

Valori logice:

```
ORD(TRUE) = 1  asa incit TRUE este reprezentat prin 1,
ORD(FALSE) = 0 asa incit FALSE este reprezentat prin 0.
```

### A.3.1.3 Numere reale.

Se foloseste formatul cu mantisa si exponent, similar celui folosit in notatia stiintifica standard, insa cu reprezentare binara in locul celei zecimale. Exemple:

$$2 = 2*10^{0} \quad \text{sau} \quad 1.0 *2^{1}_{2}$$

$$1 = 1*10^{0} \quad \text{sau} \quad 1.0 *2^{0}_{2}$$

$$-12.5 = -1.25*10^{1} \quad \text{sau} \quad -25*2^{-1}$$

$$= -11001 *2^{-1}_{2}$$

$$= -1.1001_2 * 2^{3} \quad \text{dupa normalizare.}$$

$$0.1 = 1.0*10^{-1} \quad \text{sau} \quad \frac{1}{10} = \frac{1}{1010_2} = \frac{0.1_2}{101_2}$$

Si acum urmeaza o operatie lunga de impartire in binar:

```
         0.0001100
     101 0.10000000000000
             101
              110
              101
               1000
                101     in acest punct
                        observam ca fractia
                        este periodica
```

$$= \frac{0.1_2}{101_2} = 0.0001100_2$$

$$1.1001100*2^{-4} \quad \text{raspunsul.}$$

Cum vom folosi rezultatele de mai sus pentru a reprezenta numere in calculator ? Mai intii vom rezerva 4 octeti de memorie pentru fiecare numar real, in formatul urmator:

| semn | mantisa normalizata | | exponent | | numarul |
|---|---|---|---|---|---|
| 23 | 22 | 0 | 7 | 0 | nr.crt. bit |
| H | L | E | | D | registru |

- semnul: semnul mantisei: 1 negativ, 0 pozitiv.
- mantisa normalizata: mantisa adusa la forma 1.xxxxxx, cu bitul cel mai semnificativ (bitul 22) egal totdeauna cu 1, cu exceptia cazului cind se reprezinta zero (HL=0, DE=0).
- exponent: exponentul este in binar, in format complement fata de 2.

Astfel:

```
2     = 0 1000000 00000000 00000000 00000001  ('40,'00,'00,'01)
1     = 0 1000000 00000000 00000000 00000000  ('40,'00,'00,'00)
-12.5 = 1 1100100 00000000 00000000 00000011  ('E4,'00,'00,'03)
0.1   = 0 1100110 01100110 01100110 11111100  ('66,'66,'66,'FC)
```

Amintindu-ne ca registrele HL si DE sint folosite pentru a duce in memorie numerele reale, rezulta ca pentru a stoca

numerele de mai sus trebuie sa incarcam aceste registre in felul urmator:

```
        2  =  LD   HL,`4000
              LD   DE,`0100
        1  =  LD   HL,`4000
              LD   DE,`0000
    -12.5  =  LD   Hl,`E400
              LD   DE,`0300
      0.1  =  LD   HL,`6666
              LD   DE,`FC66
```

Ultimul exemplu arata de ce calculele cu fractii binare pot fi imprecise; 0.1 nu poate fi reprezentat exact ca fractie binara, folosind un numar finit de cifre dupa punct.

**N.B.** Numerele reale sint stocate in memorie in ordinea ED LH.

### A.3.1.4 Inregistrari si tablouri.

Inregistrarile ocupa o cantitate de memorie egala cu totalul cerut pentru componentele sale.

Tablouri: daca n = numarul de elemente din tablou, iar
s = numarul de octeti ocupati de un element,
atunci numarul de octeti ocupati de tablou este n*s.
De exemplu, ARRAY[1..10] OF INTEGER necesita 10*2=20 octeti, iar ARRAY[2..12,1..10] OF CHAR are 11*10=110 elemente si necesita 110 octeti.

### A.3.1.5 Multimi.

Multimile sint stocate ca siruri de biti si deci daca tipul de baza are n elemente numarul de octeti folosit este (n-1)DIV8+1. Exemple:

SET OF CHAR necesita (256-1)DIV8+1 = 32 octeti;
SET OF (blue,green,yellow) necesita (3-1)DIV8+1 = 1 octet.

### A.3.1.6 Indicatori.

Indicatorii ocupa 2 octeti, care contin adresa variabilei dinamice in format Intel, adica cu octetul cel mai putin semnificativ in fata.

### A.3.2 Stocarea variabilelor in timpul executarii programului.

Sint 3 cazuri in care utilizatorul are nevoie de informatii cu privire la modul cum sint stocate variabilele in timpul executarii programului:

- **a. Variabile globale** - declarate in blocul principal al programului.
- **b. Variabile locale** - declarate intr-un bloc intern.
- **c. Parametri si valori returnate** - transmise catre, sau de la proceduri si functiuni.

Aceste cazuri individuale sint discutate mai jos, iar un exemplu

de modul cum sint utilizate aceste informatii poate fi gasit in Anexa 4.

**Variabile globale.**

Variabilelor globale le este alocat spatiu in stiva de executie, incepind de sus si mergind in jos. De exemplu, daca stiva de executie este la `B000, iar variabilele principale ale programului sint:

```
VAR  i:INTEGER;
     ch:CHAR;
     x:REAL;
```

atunci:

i (care ocupa 2 octeti - vezi sectiunea precedenta) va fi stocat in locatiile `B000 - 2 si `B000 - 1, adica in `AFFE si `AFFF.

ch (1 octet) va fi stocat in locatia `AFFE - 1, adica `AFFD.

x (4 octeti) va fi plasat in locatiile `AFF9, `AFFA, `AFFB, `AFFC.

**Variabile locale.**

Variabilele locale nu pot fi regasite foarte usor in stiva, dar, in schimb, se foloseste pentru aceasta registrul IX. La inceputul fiecarui bloc intern, (IX-4) indica inceputul blocului variabilelor locale. De exemplu, in cazul:

```
PROCEDURE  test;
VAR          i,j:INTEGER;
```

i (intreg pe 2 octeti) va fi plasat la adresele date de IX-4-2 si IX-4-1, adica de IX-6 si IX-5.

j va fi plasat la adresele date de IX-8 si IX-7.

**Parametri si valori returnate.**

Parametrii valori sint tratati la fel ca variabilele locale si, la fel ca acestea cu cit un parametru este declarat mai devreme cu atit mai mare este adresa din memorie la care este stocat. Spre deosebire insa, de variabilele locale, nu adresa cea mai mare, ci cea mai mica este fixata, la (IX+2). In exemplul de mai jos:

```
PROCEDURE test(i:REAL;j:INTEGER);
```

j (se aloca primele locatii) este stocat la IX+2 si IX+3.
i este stocat la IX+4, IX+5, IX+6 si IX+7.

Parametrii variabili sint tratati la fel ca parametrii valori, cu exceptia faptului ca lor le sint alocati totdeauna 2 octeti, acesti octeti continind adresa variabilei. De exemplu:

```
PROCEDURE test(i:INTEGER; VAR x:REAL);
```

referirea la x este plasata la IX+2 si IX+3; aceste locatii contin adresa unde este stocata x. Valoarea lui i este la IX+4

si IX+5.

Valorile returnate de functiuni sint plasate deasupra primului parametru. De exemplu:
FUNCTION test(i:INTEGER):REAL;

i se gaseste la IX+2 si IX+3, iar pentru valoarea returnata se rezerva spatiu la IX+4, IX+5, IX+6 si IX+7.

## ANEXA 4 EXEMPLE DE PROGRAME SCRISE IN HP4T.

Studiati cu atentie programele care urmeaza daca aveti nelamuriri asupra modului cum se programeaza in HP4T.

*

```
{Program pentru ilustrarea utilizarii functiunilor TIN si
 TOUT. Programul construieste o lista cu numere de telefon,
 foarte simpla, pe care o salveaza pe caseta si apoi o
 citeste. Dvs. trebuie sa scrieti un program de cautare in
 aceasta lista.}

PROGRAM TAPE;

CONST
  Size=10;          {Notati ca 'Size' foloseste litere mari
                             si mici, nu este 'SIZE'}
TYPE
  Entry = RECORD
            Name : ARRAY [1..10] OF CHAR;
            Number : ARRAY [1..10] OF CHAR
          END;

VAR
  Directory : ARRAY [1..Size] OF Entry;
  I : INTEGER;

BEGIN

{Crearea listei}

  FOR I:= 1 TO Size DO
  BEGIN
    WITH Directory[I] DO
    BEGIN
      WRITE('Name please');
      READLN;
      READ(Name);
      WRITELN;
      WRITE('Number please');
      READLN;
      READ(Number);
      WRITELN
```

```
    END
  END;

{Pentru a salva lista pe caseta, folositi ..}

  TOUT('Director',ADDP(Directory),SIZE(Directory));

{Pentru a citi tabloul, executati ..}

  TIN('Director',ADDR(Directory))

{Acum puteti prelucra lista asa cum doriti ..}

END.
```

*

```
{Program pentru listarea liniilor unui fisier in ordine
 inversa. Demonstreaza modul de folosire a indicatorilor,
 inregistrarilor si a functiunilor MARK si RELEASE.}

PROGRAM ReverseLine;

TYPE elem=RECORD           {Creeaza o structura de lista}
        next: ^elem;                          {inlantuita.}
        ch: CHAR
     END;
   link=^elem;

VAR prev,cor,heap: link; {Toti sint indicatori catre 'elem'}

BEGIN
 REPEAT                   {Repeta aceasta de mai multe ori.}
   MARK(heap);  {Atribuie variabilei 'heap' valoarea primei}
                {adrese a zonei variabilelor dinamice.}
   prev:=NIL;             {Nu contine inca nici o adresa.}
   WHILE NOT EOLN DO
     BEGIN
       NEW(cor);      {Creeaza o noua inregistrare dinamica}
       READ(cor^.ch);      {si ataseaza cimpul acesteia unui}
                           {caracter din fisier.}
       cor^.next:=prev;   {Acest indicator de cimp adreseaza}
       prev:=cor          {inregistrarea anterioara.}
     END;

{Scrie o linie in ordine inversa prin baleierea  in  ordine}
{inversa a inregistrarilor constituite.}

   cor:=prev;
   WHILE cor <> NIL DO     {NIL este prima}
     BEGIN
       WRITE(cor^.ch);    {Scrie acest cimp, adica caracter.}
       cor:=cor^.next     {Adresa cimpului anterior.}
     END;
   WRITELN;
```

```
    RELEASE(heap);             {Elibereaza spatiu pentru}
                               {variabile dinamice.}
    READLN                     {Asteapta o noua linie.}
  UNTIL FALSE                  {Folositi CC pentru a iesi.}
END.
```

*

```
{Program pentru demonstrarea recursivitatii.}

PROGRAM FACTOR;

{Acest program calculeaza factorialul unui  numar  introdus}
{de la claviatura 1) folosind recursivitatea si 2) printr-o}
{metoda iterativa.}

TYPE
  POSINT = 0..MAXINT;

VAR
  METHOD : CHAR;
  NUMBER : POSINT;

{Algoritmul recursiv.}

FUNCTION RFAC(N : POSINT) :INTEGER;

  VAR F : POSINT;

  BEGIN
    IF N>1 THEN F:=N * RFAC(N-1)      {RFAC invocat de N ori.}
      ELSE F:=1;
    RFAC := F
  END;

{Solutia iterativa.}

FUNCTION IFAC(N : POSINT) : INTEGER;

  VAR I,F : POSINT;
  BEGIN
    F := 1;
    FOR I := 2 TO N DO F := F*I;     {Bucla simpla.}
    IFAC := F
  END;

BEGIN
  REPEAT
    WRITE('Give method (I or R) and number    ');
    READLN;
    READ(METHOD,NUMBER);
   IF METHOD = 'R'
     THEN WRITELN(NUMBER,'! = ',RFAC(NUMBER))
     ELSE WRITELN(NUMBER,'! = ',IFAC(NUMBER))
  UNTIL NUMBER=0
```

```
480 END.
```

*

```
 10 {Program care demonstreaza cum ne putem  'murdari  miinile'}
 20 {adica cum putem modifica variabilele Pascal folosind codul}
 30 {masina. Se utilizeaza PEEK, POKE, ADDR si INLINE.}
 40
 50 PROGRAM divmult2;
 60
 70 VAR r:REAL;
 80
 90 FUNCTION divby2(x:REAL):REAL;      {Functiune pentru
100                                     impartirea rapida cu 2.}
110 VAR i:INTEGER;
120 BEGIN
130    i:=ADDR(x)+1;                   {Adreseaza exponentu lui x.}
140    POKE(i,PRED(PEEK(i,CHAR)));     {Descreste exponentul lui x
150                                     -vezi Anexa 3.1.3.}
160    divby2:=x
170 END;
180
190 FUNCTION multby2(x:REAL):REAL;     {Functiune pentru
200                                     inmultirea rapida cu 2.}
210 BEGIN
220    INLINE(`DD,`34,3);              {INC(IX+3) - exponentul lui x
230                                     -vezi Anexa 3.2.}
240    multby2:=x
250 END;
260
270 BEGIN
280    REPEAT
290      WRITE('Introduceti numarul r ');
300      READ(r);                      {Nu este  necesar  READLN
310                                     -vezi Sectiunea 2.3.14.}
320
330      WRITELN('r impartit la 2 este ',divby2(r):7:2);
340      WRITELN('r inmultit cu 2 este ',multby2(r):7:2)
350    UNTIL r=0
360 END.
```

## ANEXA 5 NOTA DE IMPLEMENTARE A HISOFT PASCAL 4TM PE SPECTRUM 48K.

### A.5.1 Incarcarea HP4TM de pe banda.

Incarcarea HP4TM se face cu comanda LOAD "". Mai intii se incarca un scurt program in BASIC, cu autostart, care va incarca efectiv blocul HP4TM. Daca este detectata o eroare de incarcare, opriti casetofonul, reinfasurati banda, faceti NEW si incarcati din nou programul cu LOAD"".
Dupa ce HP4TM s-a incarcat, el va fi lansat automat in executie

si va afisa mesajul 'Top of RAM?'. Consultati acum Sectiunea 0.0 din Manualul de programare Pascal 4T Hisoft pentru detalii privind modul de actionare in continuare.

### A.5.2 Implementarea pe SPECTRUM.

ZX SPECTRUM este un calculator destul de deosebit si intr-o oarecare masura documentatia pentru HP4TM reflecta acest lucru. Diferitele comenzi mentionate in Manualul de programare se obtin la SPECTRUM astfel:

| | |
|---|---|
| **RETURN** | prin tasta 'ENTER'. |
| CC | prin CAPS SHIFT si 1. |
| CH | DELETE, adica CAPS SHIFT si 0. |
| CI | prin CAPS SHIFT si 8. |
| CP | prin CAPS SHIFT si 3, permitind 'L'istarea textului pe printer. |
| CX | prin CAPS SHIFT si 5. |
| CS | prin CAPS SHIFT si SPACE. |

Cuvintele cheie utilizate de SPECTRUM nu sint acceptate (ceea ce consideram ca este un avantaj), intreg textul trebuind sa fie introdus folosind tastatura alfanumerica normala. Folosind SYMBOL SHIFT simultan cu alta tasta (exceptind 1) va fi generat totdeauna simbolul ASCII asociat acestei taste si nu cuvintul cheie. De ex., SYMBOL SHIFT T va genera '>', iar SYMBOL SHIFT 9 va da ')'. Nu este acceptata introducerea simbolurilor compuse <=, <> si >= decit prin combinarea simbolurilor simple < , > si =.

La pornire, editorul lucreaza cu majuscule, dar se poate trece in modul normal de lucru, cu litere mici, folosind CAPS SHIFT 2.

Aveti controlul asupra atributelor temporare ale diferitelor caractere de pe ecran prin folosirea codurilor standard de control (de exemplu WRITE(CHR(17),[.?.] va face "paper" verde), insa nu puteti modifica atributele permanente. Daca, prin utilizarea acestor coduri de control de baza, este detectata o secventa nevalabila, se afiseaza mesajul '[.?.] error', iar executia va fi oprita. De notat ca unele CHR [.?.] sint interpretate de catre HP4T (de exemplu, CHR(8) este considerat drept DELETE) si astfel aceste coduri [.?.] trecute direct catre ROM-ul Spectrum-ului. Utilizati procedura SPOUT (pag. 58) daca vreti ca aceste coduri CHR sa nu fie interpretate de HP4T.

Cind salvati text sau cod obiect pe caseta, trebuie sa fiti atenti ca inainte de a incepe salvarea, casetofonul sa fie pregatit pentru inregistrare.

Daca ati folosit comanda 'T' pentru a salva codul obiect [.?.], pentru a incarca ulterior acest program folositi comanda BASIC LOAD "" CODE. Pentru a executa programul folositi comanda BASIC RANDOMIZE USR 24608.

Din BASIC puteti reveni in editorul HP4TM fie cu RANDOMIZE USR 24603 si in acest caz textul Pascal sursa se pastreaza ('pornire calda'), fie cu RANDOMIZE USR 24598, cind textul Pascal existent se sterge ('pornire rece').

Printerul ZX este activat prin optiunea 'P' a compilatorului (vezi Manualul de programare, Sectiunea 3), sau prin CHR(16) intr-o instructiune WRITE, WRITELN. Retineti ca nu puteti utiliza CHR(16) intr-o instructiune WRITE(LN) pentru a specifica culoarea INK, dar puteti folosi CHR(15) pentru a stabili INK.

Pentru a face o copie de pe HP4TM se procedeaza in felul urmator

1. Se incarca HP4TM de pe caseta si se raspunde in mod normal la mesajele 'Top of RAM?' etc.

2. Se revine in BASIC folosind comanda 'B' a editorului.

3. Se da comanda pentru salvarea programului HP4TM pe caseta:
   SAVE "HP4T15M" CODE 24598,19558

4. Puteti apoi incarca compilatorul cu LOAD "" CODE , dar retineti ca el poate fi lansat in executie numai cu RANDOMIZE USR 24598 ('pornire rece'), sau RANDOMIZE USR 24603 ('pornire calda ').

Sinteti autorizat de Hisoft faceti o singura copie de lucru.

## ANEXA 6 SUNET SI GRAFICA IN ZX SPECTRUM FOLOSIND PASCAL HP4T.

In aceasta anexa sint date amanunte cu privire la controlul sunetului si al posibilitatilor grafice din ZX SPECTRUM folosind Pascal 4T Hisoft.

### A.6.1 Sunetul.

Sint necesare urmatoarele doua proceduri (definite in ordinea de mai jos), pentru a produce sunet cu HP4T.

```
{Aceasta procedura foloseste o rutina in cod masina pentru a
 fixa parametrii si a-i trece apoi rutinei BEEP din ROM-ul
 SPECTRUM-ului.}

PROCEDURE BEEPER (A, B : INTEGER);

 BEGIN
  INLINE(#DD,#6E,2,#DD,#66,3,        {LD  L,(IX+2) : LD  H,(IX+3)}
         #DD,#5E,4,#DD,#56,5,        {LD  E,(IX+4) : LD  D,(IX+5)}
         #CD,#B5,3,#F3)              {CALL #3B5 : DI}
```

```
END;

{Aceasta procedura converteste perioadele cu frecventa zero in
perioade de liniste. Daca frecventa este diferita de zero,
frecventa si durata notei sint aproximativ convertite in valori
cerute de rutina din ROM si aceasta este apelata prin BEEPER.}

PROCEDURE BEEP (Frequency : INTEGER; Length : REAL);

VAR I : INTEGER;

 BEGIN

  IF Frequency=0 THEN FOR I:=1 TO ENTIER(12111*Length) DO

  ELSE BEEPER(ENTIER(Frequency*Length),ENTIER(437500/Frequency-
                                                       30.125)
  FOR I:=1 TO 100 DO      {o scurta pauza intre note}

 END;
```

Un exemplu de utilizare a procedurii BEEP:

```
BEEP(262,0.5);         {Nota do din octava de mijloc, urmata de}
BEEP(0,1)              {o pauza de o secunda.}
```

## A.6.2 Grafica.

Sint date trei proceduri grafice: prima este echivalenta cu comanda PLOT X,Y din BASIC si serveste pentru a marca pozitia curenta, iar celelalte doua sint utilizate pentru a trasa linii de la pozitia curenta la o pozitie noua, definita in raport cu cea curenta si care devine apoi pozitie curenta.

Atit procedura PLOT, cit si LINE au ca parametru variabila logica 'ON'; daca 'ON' este TRUE, punctul va fi marcat oricare ar fi starea pixel-ului in acea pozitie, iar daca 'ON' este FALSE starea pixel-ului din pozitia trasata va fi inversata, efect identic cu cel provocat de comanda OVER din SPECTRUM BASIC.

```
{O procedura care dubleaza comanda PLOT X,Y din BASIC, pixel-ul
 X,Y fiind activat sau nu in functie de valoarea TRUE/FALSE a
 primului parametru.}

PROCEDURE PLOT(ON : BOOLEAN; X, Y : INTEGER);

 BEGIN
 IF ON THEN WRITE(CHR(21),CHR(0))
      ELSE WRITE(CHR(21),CHR(1));

 INLINE(#FD,#21,#3A,#5C,        {LD   IY,#5C3A}
        #DD,#46,2,#DD,#4E,4,    {LD   B,(IX+2) : LD   C,(IX+4)}
```

```
        #CD,#E5,#22)                {CALL #22E5 ;rutina PLOT din ROM}
END;

{Apelata de procedura LINE, LINE1 este utilizata pentru a
 transmite rutinei DRAW din ROM argumentele corecte.}

PROCEDURE LINE1(X,Y,SX,SY ; INTEGER);

 BEGIN
  INLINE(#FD,#21,#3A,#5C,           {LD   IY,#5C3A}
         #DD,#56,2,#DD,#5E,4,       {LD   D,(IX+2) : LD   E,(IX+4)}
         #DD,#46,6,#DD,#4E,8,       {LD   B,(IX+6) : LD   C,(IX+8)}
         #CD,#BA,#24)               {CALL #24BA ;rutina DRAW din ROM}
 END;

{LINE traseaza o linie de la pozitia curenta (x,y) la (X+x,Y+y).
 Linia poate fi 'on' sau 'off', in functie de valoarea
 parametrului logic ON.}

PROCEDURE LINE(ON : BOOLEAN; X, Y : INTEGER);

VAR
  SGNX, SGNY : INTEGER;

 BEGIN
  IF ON THEN WRITE(CHR(21),CHR(0))
       ELSE WRITE(CHR(21),CHR(1));

  IF X<0 THEN SGNX:=-1 ELSE SGNX:=1;   {Rutina DRAW care urmeaza}
  IF Y<0 THEN SGNY:=-1 ELSE SGNY:=1;   {a fi apelata necesita }
             {valorile absolute si semnele parametrilor X,Y.}
  LINE1(ABS(X), ABS(Y), SGNX, SGNY)  {Trasarea liniei.}
 END;
```

Exemplu de utilizare a procedurilor PLOT si LINE:

```
PLOT(ON, 50, 50);            {Traseaza dreapta de la (50,50) }
LINE(ON, 100, -50);          {la (150,0).}
```

## A.6.3 Iesire (scriere) prin rutinele din ROM.

Sint ocazii cind este util sa iesim direct prin rutina RST #10 din ROM-ul SPECTRUM-ului, in loc sa folosim WRITE(LN). De exemplu, cind folosim codul de control PRINT AT, acesta trebuie sa fie urmat de doua valori pe 8 biti, care dau coordonatele X,Y ale punctului in care se schimba pozitia de afisare. Daca insa utilizam instructiunea Pascal WRITE, unele valori ale lui X si Y nu vor fi transmise spre ROM (de exemplu, 8 va fi interpretat de HP4T drept BACKSPACE), asa incit pozitia de afisare nu va fi modificata corect.

Putem depasi aceasta problema folosind procedura urmatoare:

```
{SPOUT afiseaza caracterul folosit ca parametru direct prin
```

rutina RST #10 din ROM, evitind interpretarea eronata de catre HP4T a parametrilor de iesire.}

```
PROCEDURE SPOUT ( C : CHAR );

BEGIN
 INLINE(#FD, #21, #3A, #5C,          {LD  IY,#5C3A}
        #DD, #7E, 2,                 {LD  A,(IX+2)}
        #D7 )                        {RST #10)}
END;
```

Exemplu de utilizare a procedurii SPOUT:

```
SPOUT ( CHR(22) ); SPOUT ( CHR(8) ); SPOUT ( CHR(13) );
           {stabileste pozitia de afisare in punctul avind
            linia 8 si coloana 13.}
```

NOTA. In aceasta anexa am folosit in locul semnului '`', pentru a marca numerele hexazecimale, semnul '#' ( CHR(35), SHIFT 3).

## ANEXA 7 GRAFICA CU CURSOR ("BROASCA") PENTRU ZX SPECTRUM IN CADRUL PASCAL 4 HISOFT.

Pascal 4T Hisoft pentru ZX SPECTRUM este livrat incepind de la 15 august 1983, cu un program suplimentar de grafica in stil LOGO, inregistrat pe partea B a casetei.

Programul este scris in Pascal si poate fi incarcat din editorul HP4T, folosin comanda 'G,,TURTLE'. In acest fel va fi incarcat segmentul de program pentru grafica si va fi adaugat la orice program existent. De notat ca, pentru a functiona corect, grafica "BROASCA" trebuie sa fie precedata de un titlu PROGRAM normal si o declaratie VAR. Declaratiile TYPE, CONST si LABEL sint optionale; de asemenea, nu trebuie sa existe proceduri si functiuni declarate inainte de includerea graficii "BROASCA".

Ca in majoritatea implementarilor cu grafica "BROASCA" si cea din Pascal Hisoft arata pe ecran o creatura imaginara, care poate fi deplasata cu ajutorul unor comenzi foarte simple. Aceasta "broasca" poate fi facuta sa lase o urma, o coada (de diferite culori), sau poate fi facuta invizibila. Pozitia si orientarea "broastei" se pastreaza in variabile globale, care sint actualizate cind creatura este deplasata sau rotita; evident, aceste variabile pot fi inspectate si modificate in orice moment.

Facilitatile disponibile sint urmatoarele:

### A.7.1 Variabile globale.

**HEADING**

Este utilizata pentru a pastra valoarea unghiulara a orientarii "broastei". Poate lua orice valoare REALa, in grade si poate fi initializata la zero prin procedura TURTLE (vezi mai jos). Valoarea 0 corespunde directiei EST, asa incit dupa o apelare a procedurii TURTLE "broasca" va privi spre dreapta. Cind variabila HEADING creste, "broasca" se va roti in sens invers acelor unui ceasornic.

**XCOR, YCOR**

Acestea sint coordonatele curente (x,y) REALe ale "broastei" pe ecran. Ecranul SPECTRUM-ului are 256*176 pixeli si "broasca" poate fi pozitionata in oricare punct al acestei suprafete; daca se incearca sa se scoata "broasca" din aceasta arie (folosind LINE - vezi mai jos), va fi afisat mesajul 'Out of Limits', iar programul va fi oprit cu un mesaj 'HALT'. La inceput XCOR si YCOR sint nedefinite, apelarea procedurii TURTLE le initializeaza la valorile 127, respectiv 87, adica plaseaza "broasca" in mijlocul "baltii" sale.

**PENSTATUS**

O variabila de tip intreg care pastreaza starea curenta a "penitei" (adica a urmei lasate de "broasca"). 0 inseamna "penita" jos ("broasca" lasa urma), iar 1 "penita" sus (nu lasa urma).

## A.7.2 Proceduri.

Procedurile PLOT,LINE si SPOUT descrise in Anexa 6 sint incluse in acest program. Singura modificare importanta in implementarea de aici este ca LINE apeleaza o procedura CHECK, care asigura ca XCOR si YCOR nu pot depasi limitele ecranului.

Celelalte proceduri sint:

**INK ( C : INTEGER )**

C este un intreg intre 0 si 8 inclusiv; procedura stabileste culoarea "cernelii", a urmei lasate de "broasca".

**PAPER ( C : INTEGER )**

Stabileste culoarea fondului ecranului ("hirtiei"), conform culorii asociate parametrului C, un intreg intr 0 si 8 inclusiv.

**COPY**

Trimite imaginea curenta de pe ecran pe printerul ZX; este utila pentru a salva pe printer o pagina completata de grafica.

**PENDOWN ( C : INTEGER )**

Modifica starea "broastei", astfel incit ea sa lase o urma avind culoarea asociata parametrului C, care este un intreg intre 0 si 8 inclusiv. Aceasta procedura face 0 variabila PENSTATUS.

**PENUP**

Dupa apelarea acestei proceduri, "broasca" nu va mai lasa urma. Utila pentru deplasari dintr-o sectiune in alta a graficii.

PENUP face PENSTATUS = 1.

**SETHD ( A : REAL )**
A este un parametru real care este atribuit variabilei globale HEADING, stabilind astfel orientarea "broastei". Retineti ca o valoare 0 a acestei variabile corespunde unei orientari spre EST 90 spre NORD, 180 spre VEST si 270 spre SUD.

**SETXY ( X, Y : REAL )**
Stabileste pozitia absoluta a "broastei" pe ecran in punctul (X,Y). In cadrul acestei proceduri nu se verifica daca punctul (X,Y) nu depaseste limitele ecranului. Procedura LINE este cea care executa acest control.

**FWD ( L : REAL )**
Deplaseaza "broasca" inainte cu L unitati, in directia in care este orientata. O unitate corespunde unui pixel grafic, rotunjit daca este cazul in sus sau in jos.

**BACK ( L : REAL )**
Deplaseaza "broasca" cu L unitati in directia opusa orientarii ei curente (adica la -180). Orientarea "broastei" ramine neschimbata.

**TURN ( A : REAL )**
Modifica orientarea "broastei" cu A grade, fara sa o deplaseze. Orientarea este marita in sens trigonometric.
VECTOR ( A,L : REAL )
Deplaseaza "broasca" cu L unitati in directia data de A. Dupa deplasare, "broasca" ramine cu orientarea initiala.

**RIGHT ( A : REAL )**
Ca alternativa la TURN, RIGHT modifica orientarea "broastei", dar in sensul acelor de ceas, cu A grade.

**LEFT ( A : REAL )**
Procedura este identica cu TURN si este data pentru comoditate si pentru compatibilitate cu RIGHT.

**ARCR ( R : REAL, A : INTEGER )**
"Broasca" se va deplasa pe un arc de cerc cu raza R. Lungimea arcului este determinata de A, unghiul la centru in sens orar. Tipic R ia valoarea 0.5.

**TURTLE**
Aceasta procedura initializeaza starea "broastei", plasind-o in mijlocul ecranului, cu fata spre EST, pe un fond albastru si lasind o urma galbena. Retineti ca starea "broastei" nu este definita initial, asa incit aceasta procedura este de obicei utilizata la inceputul programului.

Cu aceasta se incheie lista facilitatilor disponibile in cadrul HP4T TURTLE. Desi simplu ca implementare si utilizare, veti descoperi ca programul de grafica "Broasca" poate produce desene foarte complexe, cu viteza mare. Pentru a va demonstra acest

lucru, va prezentam in continuare citeva exemple de programe. Retineti ca, inainte de a introduce aceste programe, trebuie sa fi incarcat Pascal 4T Hisoft.

## Exemple de programe.

In toate exemplele de programe date mai jos, presupunem ca ati incarcat deja Pascal 4T Hisoft si apoi cu comanda 'G,,TURTLE' ati incarcat programul de grafica "Broasca", care incepe la linia 10 si se termina la linia 1350. Si acum sa continuam cu exemplele:

### 1. CERCURI

```
   1 PROGRAM CIRCLES;
   2 VAR I:INTEGER;

1360 BEGIN
1370   TURTLE;
1380   FOR I:=1 TO 9 DO
1390   BEGIN
1400    ARCR(0.5,360);
1410    RIGHT(40)
1420   END
1430 END.
```

### 2. SPIRALE

```
   1 PROGRAM SPIRALS;
   2 VAR

1360 PROCEDURE SPIRALS (L,A;REAL );
1370 BEGIN
1380  FWD(L);
1390  RIGHT(A);
1400  SPIRALS(L+1,A);
1410 END;
1420 BEGIN
1430  TURTLE;
1440  SPIRALS(9,95)                  (sau (9,90), (9,121) . . .)
1450 END.
```

### 3. FLOARE

```
   1 PROGRAM FLOWER;
   2 VAR

1360 PROCEDURE PETAL ( S:REAL );
1370 BEGIN
1380  ARCR(S,60);
1390  LEFT(120);
1400  ARCR(S,60);
```

```
1410  LEFT(120)
1420 END;
1430 PROCEDURE FLOWER ( S:REAL );
1440 VAR I:INTEGER;
1450 BEGIN
1460  FOR I:=1 TO 6 DO
1470  BEGIN
1480   PETAL(S);
1490   RIGHT(60)
1500  END
1510 END;
1520 BEGIN TURTLE;
1530  SETXY(127,60);
1540  LEFT(90);  FWD(10);
1550  RIGHT(60); PETAL(0.2);
1560  LEFT(60);  PETAL(0.2);
1570  SETHD(90); FWD(40);
1580  FLOWER(0.4)
1590 END.
```

Pentru un studiu mai extins al graficii "Broasca" va recomandam calduros cartea "Turtle Geometry" de Harold Abelson si Andrea di Sessa, publicata de MIT Press, ISBN 0-262-01063-1.

**NOTA IMPORTANTA**

Hisoft regreta ca s-a strecurat o mica eroare in procedura COPY livrata cu programul TURTLE. In prezent, linia 570 a acestui program este 'USER (#EAC) si ea trebuie modificata in felul urmator:

```
INLINE ( #ED,#21,#3A,#5C,#FD,#CB,1,#CE,#CD,#AC,#E,
         #FD,#CB,1,#8E,#F3).
```

## ANEXA 8  HISOFT PASCAL 4 - VERSIUNEA 1.4

Numarul de versiune pentru Hisoft Pascal 4D si Hisoft Pascal 4T este 1.4, incepind de la 31 octombrie 1982.

Diferentele fata de versiunile precedente ale implementarii Hisoft Pascal 4 sint urmatoarele:

1. A fost corectata eroarea care facea ca expresii de tipul 1 + SQR(2) sa fie evaluate gresit.

2. A fost corectata eroarea care facea ca expresii de tipul I + (I+1) sa fie evaluate gresit.

3. A fost corectata eroarea care ducea la evaluarea incorecta a rezultatului comparatiilor intreg-real.

4. A fost corectata eroarea mentionata in Manualul de programare (pag. 40) la subcomanda 'S'; aceasta subcomanda poate fi folosita acum oriunde.

5. Utilizind comanda 'F' a editorului pentru a gasi un sir de caractere, cursorul de editare va fi pozitionat acum la inceputul sirului gasit curent (vezi pag. 39-40 din Manual).

6. A fost incorporata o noua comanda in editor: comanda 'X' afiseaza in hexazecimal adresa curenta de sfirsit a compilatorului. Aceasta informatie permite utilizatorului sa faca o copie de lucru a compilatorului. Aceasta nu se referea insa la utlizatorii ZX SPECTRUM care pentru a face o copie de lucru de pe HP4T se vor calauzi dupa indicatiile din Nota de implementare.

## ANEXA 9 HISOFT PASCAL - VERSIUNEA 1.5.

De la 1 aprilie 1983, numarul de versiune a implementarii Hisoft Pascal 4T este 1.5. Diferentele fata de versiunea 1.4 sint urmatoarele:

1. Functiunile inapoiaza acum un rezultat de tip indicator.

2. A fost corectata o eroare la procedura predefinita NEW, care conducea uneori la alocari incorecte de memorie pentru variabile dinamice.

3. A fost incorporata o noua comanda in editor. Comanda 'V', fara argumente, afiseaza valorile implicite curente pentru domeniul de linii, sirul f si sirul s (din comanda 'F'). Domeniul implicit curent de linii este afisat primul, urmat de cele doua siruri (care pot fi si vide), pe linii separate. Retineti ca unele comenzi ale editorului (cum sint 'D' si 'N') nu folosesc aceste valori implicite si cer ca valorile sa fie specificate in linia de comanda (vezi Manualul de programare HP4T).

4. In versiunea HP4T pentru ZX SPECTRUM este disponibila acum optiunea 'F' a compilatorului (vezi Sectiunea 3.2 a Manualului de Programare HP4T). Daca vreti sa utilizati aceasta optiune, textul sursa care urmeaza sa fie inclus trebuie sa fi fost salvat cu ajutorul unei noi comenzi a editorului, comanda 'W'. Comanda 'W' functioneaza la fel ca 'P', cu exceptia faptului ca nu salveaza in format standard HP4T si a faptului ca incepe sa salveze textul pe banda imediat dupa ce linia de comanda a fost introdusa. Rezulta ca trebuie sa aveti casetofonul pregatit pentru inregistrare inainte de a apasa tasta ENTER la sfirsitul liniei de comanda. Cind se salveaza text cu ajutorul comenzii 'W', sau cind se citeste text folosind optiunea 'F' a compilatorului, apasind tasta BREAK se poate reveni in orice moment in editor.

Retineti ca daca vreti sa cititi din editor text cu ajutorul comenzii 'G', trebuie ca acest text sa fi fost salvat cu comanda 'P', nu cu 'W'.

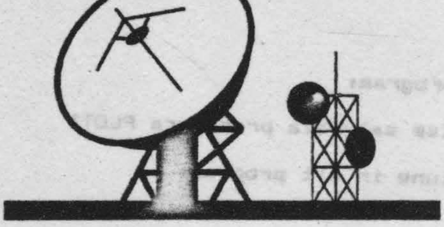
**Exemple de utilizare.**
**Pentru a salva o sectiune de program:**

```
W30,120,PLOT                    {se salveaza procedura PLOT}
```

**Pentru a include aceasta sectiune in alt program:**

```
100 DO;
110
120 {$F PLOT                    aici este inclusa procedura PLOT}
130
140 PROCEDURE MORE;            {restul programului}
150
```

**NOTA FINALA:** S-au depus toate eforturile ca materialul prezentat sa fie clar si corect. In cazul in care s-au strecurat erori de traducere sau de formulare, va rugam sa luati legatura cu **ALPHA Ltd.** Va asteptam la tel. 961/12936 sau 1900 Timisoara 5, PO Box 655 pentru Catalogul **gratuit** de oferte



## IMPORTANT !

Edituta "TM" pune la dispozitia tuturor celor interesati întreaga gamă de manuale în limba română pentru calculatoare compatibile ZX Spectrum (TIM.S, TIM S Plus, COBRA, HC 85, CIP, Jet) editate de firma **"ALPHA Ltd"** S.R.L. :

1.01 Limbajul **BASIC** pe întelesul tuturor în 12 lectii
1.02 Documentatie **GENS** și **MONS** (Asamblor-dezasamblor)
1.03 Documentatie limbaj **FORTH**
1.04 Documentatie **BETA BASIC 3.1** (Extensie BASIC)
1.05 Documentație **BETA BASIC 3.1** (Rezumat)
1.06 Documentatie compilator **FORTRAN 77-S**
1.07 Documentatie editor de texte **TASWORD**
1.08 Documentatie compilator **BLAST**
1.09 Documentatie compilator **PASCAL HP4TM** (Rezumat)
1.10 Documentatie limbaj **C**
1.11 Memento timing cod mașină **Z80**
1.12 Documentatie **MEGA BASIC** (Extensie BASIC)
1.13 Documentatie **VU-CALC**
1.14 Manual **BASIC** avansati - conținînd și referiri la **COBRA**
1.15 Documentatie compilator **COLT**
1.16 Documentatie **MASTER - FILE** (sistem gestiune afaceri)
1.17 Documentatie limbaj **microPROLOG**
1.18 Documentatie limbaj **PASCAL HP4TM**
1.19 Documentatie sistem operare **CP/M** cu referire la calculatorul **COBRA**
1.20 Manual **ROM SPECTRUM** complet dezasamblat
1.21 Documentatie **LASER GENIUS** (pachet programe pentru lucrul în cod masină)
1.22 Cum să obtinem cît mai mult de la calculatorul nostru o carte cu programe si trucuri atît pentru începători cît si pentru avansati, în două variante:
a) Numai cartea, cu o parte din figuri în text
b) Cartea si o casetă demonstrativă, cu toate programele si figurile introduse
1.23 Construiți singuri 20 de montaje electronice interfațabile cu microcalculatorul Dvs

# SOCIETATEA COMERCIALĂ "TM" S.R.L.

- **editează si tipăreste**
  - revista de "kit"-uri si informatii în electronică **"RET"**
  - suplimente, cataloage, cărti în domeniul tehnicii de calcul si electronicii
- **produce** "kit"-uri în electronică
- **execută** comenzi de producător pe bază de contract cu orice beneficiar
- **comercializează** - prin magazine proprii, retea proprie de distributie în tară, coletărie, mesagerie sau livrare directă cu mijloace auto:
  - toate publicatiile periodice sau neperiodice din domeniul de activitate, produse în tară;
  - componente active ale **S.C. "MICROELECTRONICA" S.A.** din Bucuresti: integrate MOS, integrate speciale, componente optoelectronice;
  - conectică produsă de **"CONECT" S.A.** Bucuresti: intrerupătoare, conectoare, mufe, cabluri, etc;
  - componente pasive realizate de **"IPEE"** Curtea de Arges: rezistente cu peliculă de carbon, peliculă metalică sau bobinate, condensatoare ceramice, multistrat sau de trecere, potentiometre si semireglabile, trimeri, sonerii, relee de semnalizare, etc;
  - relee, temporizatoare si transformatoare de putere mică produse de **"RELEE"** Medias;
  - ferite diverse realizate de **"Aferro"** Bucuresti;
  - borne, izolatori plastic, sonde osciloscop, aparatură diversă produse de **"ICE"** Bucuresti;
  - generatoare de miră color, convertoare PAL, aparatură complexă antifurt realizate de **"ROEL"** Bucuresti;
  - casete cu jocuri si programe, diverse cărti de informatică realizate de **"ALPHA Ltd"** Timisoara;
  - piese de schimb radio-Tv;
  - componente diverse aflate în consignatie sau aduse din import;
  - diskete si consumabile pentru calculatoare.

**Vă rugăm să ne contactati pe adresa 1900 Timisoara, str. Miron Costin Nr. 2, Telefon 96/11.85.76 .**

RET

5. Utilizînd comanda 'F' a editorului pentru a găsi un șir de caractere, cursorul de editare va fi poziționat acum la începutul șirului găsit (vezi pag. 57-60 din Manual).

6. A fost încorporată o nouă comandă în editor: comanda 'X' afișează în hexazecimal adresa curentă de sfîrșit a compilatorului. Această informație permite utilizatorului să facă o copie de lucru a compilatorului. Rețineți că se referă doar la utilizatorii ZX SPECTRUM care pentru a face o copie de lucru de pe [?] se vor conduce după indicațiile din Nota de Implementare.

## ANEXA 5 HISOFT PASCAL - VERSIUNEA 1.5.

La 1 aprilie 1983, numărul de versiune a implementării HiSoft Pascal 4T este 1.5. Diferențele față de versiunea 1.4 sînt următoarele:

1. Funcțiunile logaritmice sînt cu rezultat de tip indicator.

2. A fost corectată o eroare la procedura predefinită NEW, care conducea uneori la alocări incorecte de memorie pentru variabile dinamice.

3. A fost încorporată o nouă comandă în editor: Comanda 'V', fără argumente, afișează valorile implicite curente pentru domeniul de linii, șirul 'f' și șirul 's' (vezi comanda 'F'). Numerele implicite curente de linie sînt afișate primul, urmat de cele două șiruri (care pot fi și vide), pe linii separate. Rețineți că unele comenzi ale editorului (cum sînt 'D' și 'N') nu folosesc aceste valori implicite ci cer ca valorile să fie specificate în linia de comandă (vezi Manualul de programare HP4T).

4. În versiunea HP4T pentru ZX SPECTRUM este disponibilă acum opțiunea 'F' a compilatorului (vezi Secțiunea 3.2 a Manualului de Programare HP4T). Dacă vreți să utilizați această opțiune, textul sursă care urmează să fie inclus trebuie să fi fost salvat cu ajutorul unei noi comenzi a editorului, comanda 'W'. Comanda 'W' funcționează la fel ca 'P', cu excepția faptului că nu salvează în format standard HP4T și a faptului că începe să salveze textul pe bandă imediat după ce linia de comandă a fost introdusă. Rezultă că trebuie să aveți casetofonul pregătit pentru înregistrare înainte de a apăsa tasta ENTER la sfîrșitul liniei de comandă. Cînd se salvează text cu ajutorul comenzii 'W', sau cînd se citește text folosind opțiunea 'F' a compilatorului, apăsînd tasta BREAK se poate reveni în orice moment în editor.

Rețineți că dacă vreți să citiți în editor text cu ajutorul comenzii 'G', trebuie ca acest text să fi fost salvat cu comanda 'P', nu cu 'W'.